Nivel de negocio en J2EE: Enterprise Java Beans

Pablo Basanta Val
Florina Almenares Mendoza
Basado en material de:
Natividad Martínez Madrid, Marisol García Valls y Simon Pickin
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
{pbasanta,florina,nati,mvalls,spickin}@it.uc3m.es

Objetivos didácticos

- Comprender cómo funciona la arquitectura de los EJB's dentro del conjunto de especificaciones J2EE
  - Conocer los **diferentes tipos** de EJBs existentes y relación con la lógica de negocio (Entidades y Procesos de negocio)
  - Conocer la **estructura interna** de un EJB: Interfaz Remota y el descriptor de despliegue
  - Conocer **cómo interactúan** clientes y los EJBs mediante el empleo de servicios y el contendor

- Entender los diferentes roles y herramientas que aparecen en una aplicación J2EE
  - **Desarrollador, ensamblador de aplicaciones, desplegador,** administrador, proveedor de contenedores y proveedor del servidor del servidor de EJB's

Índice (1/2)

Índice (2/2)

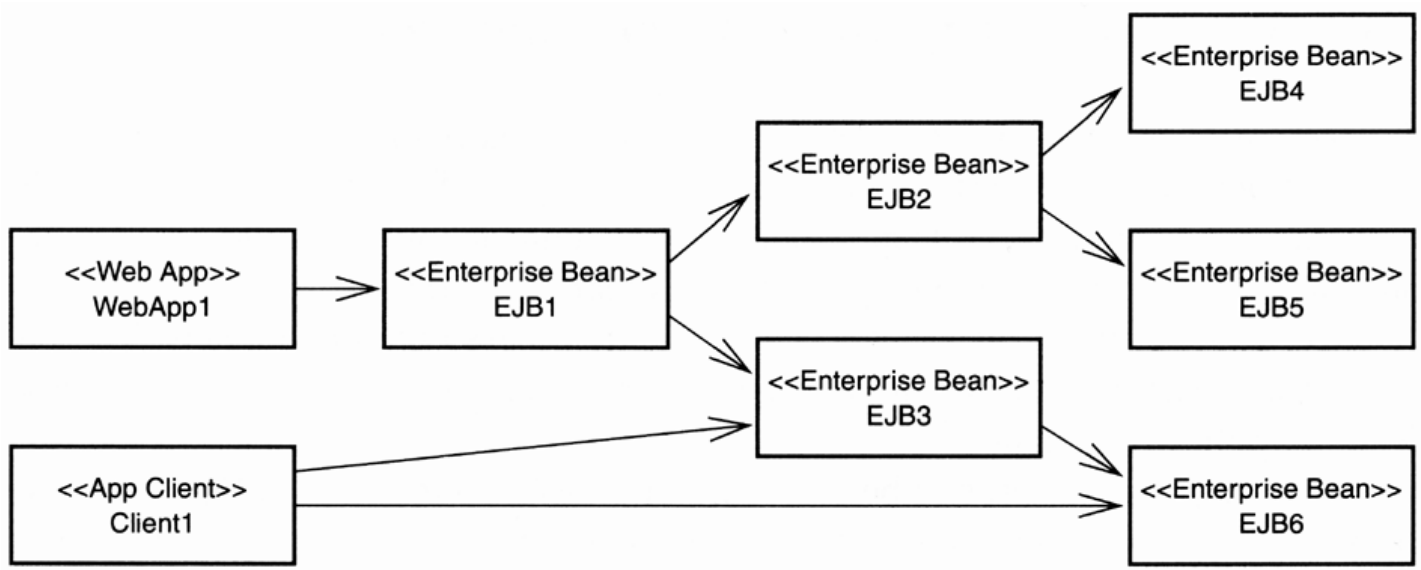
Índice (3/3)

Qué es un Enterprise Java Bean

- EJB = componente que puede ser utilizado en la construcción de aplicaciones corporativas distribuidas
- Encapsula una parte de la lógica de negocio de la aplicación
- Accede a otros gestores de recursos como bases de datos y otros enterprise beans
- Es accedido por otros enterprise beans, aplicaciones, servlets y aplicaciones cliente
- Reside en un contenedor que provee soporte cierto soporte (seguridad, transacción, despliegue, concurrencia y gestión de su ciclo de vida)
- Forma parte de una enterprise aplication

Enterprise Application

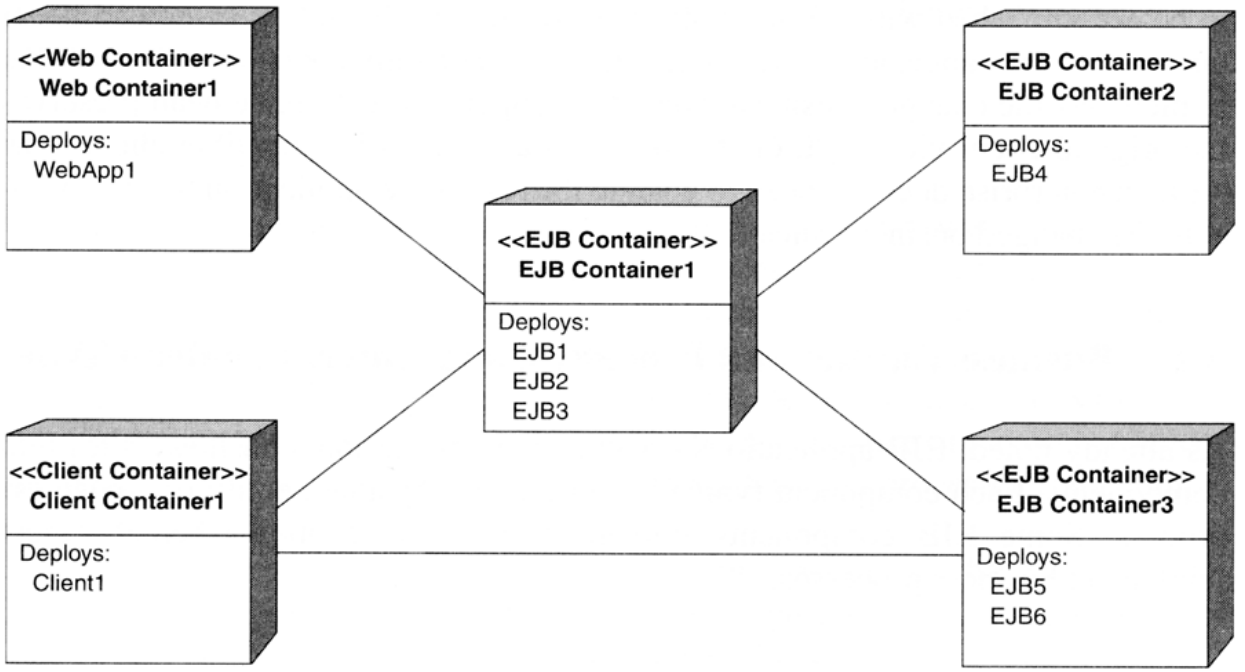
- Una enterprise aplication consta de EJBs y otros componentes (web y aplicaciones cliente)
  - En el ejemplo tenemos 6 EJBs (EJB, EJB2, EJB3, EJB4, EJB5, EJB6), un aplicación web (WebApp1) y una aplicación cliente (Client1)



Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

Enterprise Application

- Una EA puede estar desplegada en un contenedor o en varios
  - En el ejemplo, *webContainer* contiene `WebApp1`, *ClientContainer* a `Client1`, *EJBContainer1* a {`EJB1,EJB2,EJB3`}, *EJBContainer* a `EJB4` y *EJBContainer* a {`EJB5, EJB6`}



Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

La interfaz cliente del EJB

- Permite acceder a la lógica de negocio de un EJB
  - Es Independiente de la localización del EJB
    - Funciona tanto localmente como remotamente
    - Aunque el contenedor puede optimizar el acesos locales
  - Y del tipo de contener utilizado
    - Es misma para un cliente web, otro EJB o una aplicación cliente
- Es utilizado por los desarrolladores en la realización de las siguientes tareas:
  - Combinación de diferentes enterprise beans en aplicaciones multicapa
  - Construcción de aplicaciones clientes (por ejemplo aplicaciones Web)
  - Integración de EA's desarrolladas por otros vendedores
- Una buenas interfaces mejoran la adaptabilidad del sistema:
  - Cambios en las "reglas del negocio" no deberían de cambiar las interfaces

Modelado de aplicaciones empresariales en J2EE

- Las aplicaciones empresariales se organizan mediante entidades de negocio y procesos de negocio
  - Entidad de negocio = información de la empresa
  - Proceso de negocio = cómo se manipula la información

- La tecnología J2EE ofrece soporte para entidades de negocio y procesos de negocio
  - Mediante SesionBeans (para clientes Web), EntityBeans (acceso a la base de datos) y MessageBeans (procesado de mensajes entrantes)

Entidades de negocio

- Representan un objeto de negocio que contiene alguna información mantenida por una empresa
  - Típicamente estado o valores almacenados en una base de datos
  - Por ejemplo los clientes, sus compras y un código postal almacenado en una base de datos
- Las reglas de negocio asociadas a una entidad de negocio
  - Restringen su comportamiento (por ejemplo, un código postal no ha de tener más de 5 o 9 letras)
  - Mantienen relaciones entre varias entidades de negocio (por ejemplo, qué sucede con las compras pendientes de un cliente si éste desaparece)

Proceso de negocio

- Es un proceso que típicamente encapsula una interacción de un usuario con entidades de negocio.
  - Actualiza el estado de la entidad de negocio
- Puede tener su propio estado:
  - **Persistente**: proceso realizado por diferentes actores (también llamado proceso colaborativo)
    - Ejemplo: el proceso de solicitud de un préstamo o un informe gastos
  - **Transitorio**: proceso de un único actor (denominado proceso conversacional)
    - Ejemplo: la retirada de dinero de un cajero
- Muchos de los procesos de negocio de las aplicaciones web pueden ser consideradas como procesos de negocio

Reglas de negocio

- Se reparten entre los componentes que representan las diferentes entidades de negocio y los procesos de negocio
  - Dependiendo de la naturaleza de la aplicación
  - Hay reglas que típicamente se asocian a entidades de negocio
    - Saber si el saldo de una cuenta es negativo aparece ligado a la entidad Cuenta en un balance
  - Otras reglas típicamente aparecen ligadas a un proceso de negocio
    - La que impide obtener billetes de 10€ en cajeros

Lógica de negocio en J2EE

- J2EE ofrece soporte para la lógica de negocio mediante varios tipos de beans:
  - **Entity Beans**:
    - modelan conceptos de negocio como objetos persistentes asociados a datos
    - Soporte pasivo de información que ofrece métodos para las operaciones sobre los datos (ej. Cuenta bancaria, producto, pedido)
    - Llamada síncrona
  - **Session Beans**:
    - Representan procesos ejecutados en respuesta a una solicitud del cliente (ej. Transaccionesbancarias, cálculos, realización de pedidos)
    - Típicamente usan Entity Beans para el procesado de datos
    - Llamada síncrona
  - **Message-Driven Beans**:
    - Representan procesos ejecutados como respuesta a la recepción de un mensaje
    - Llamada asíncrona
- Los beans son inmutables, no se pueden cambiar de role durante el transcurso de una aplicación

Session Beans

- Realizan la parte del servidor de la lógica de negocio de una aplicación
- Cuando un cliente realiza una llamada a un Session Bean se crea una instancia (Objeto Session) asociada a ese cliente como recurso privado. El objeto Session se borra o libera al acabar el proceso del cliente
- Los hay de dos tipos:
  - **Stateless Session Beans** (sin estado):
    - No almacenan datos del cliente (sólo utilizan los datos pasados como parámetros)
    - Todos los objetos Session de un bean poseen la misma identidad
  - **Stateful Session Beans** (con estado):
    - Tienen estados dependientes del cliente
    - El estado no es persistente, se pierde cuando el objeto Session se deja de utilizar
    - Cada objeto Session tiene distintas identidades

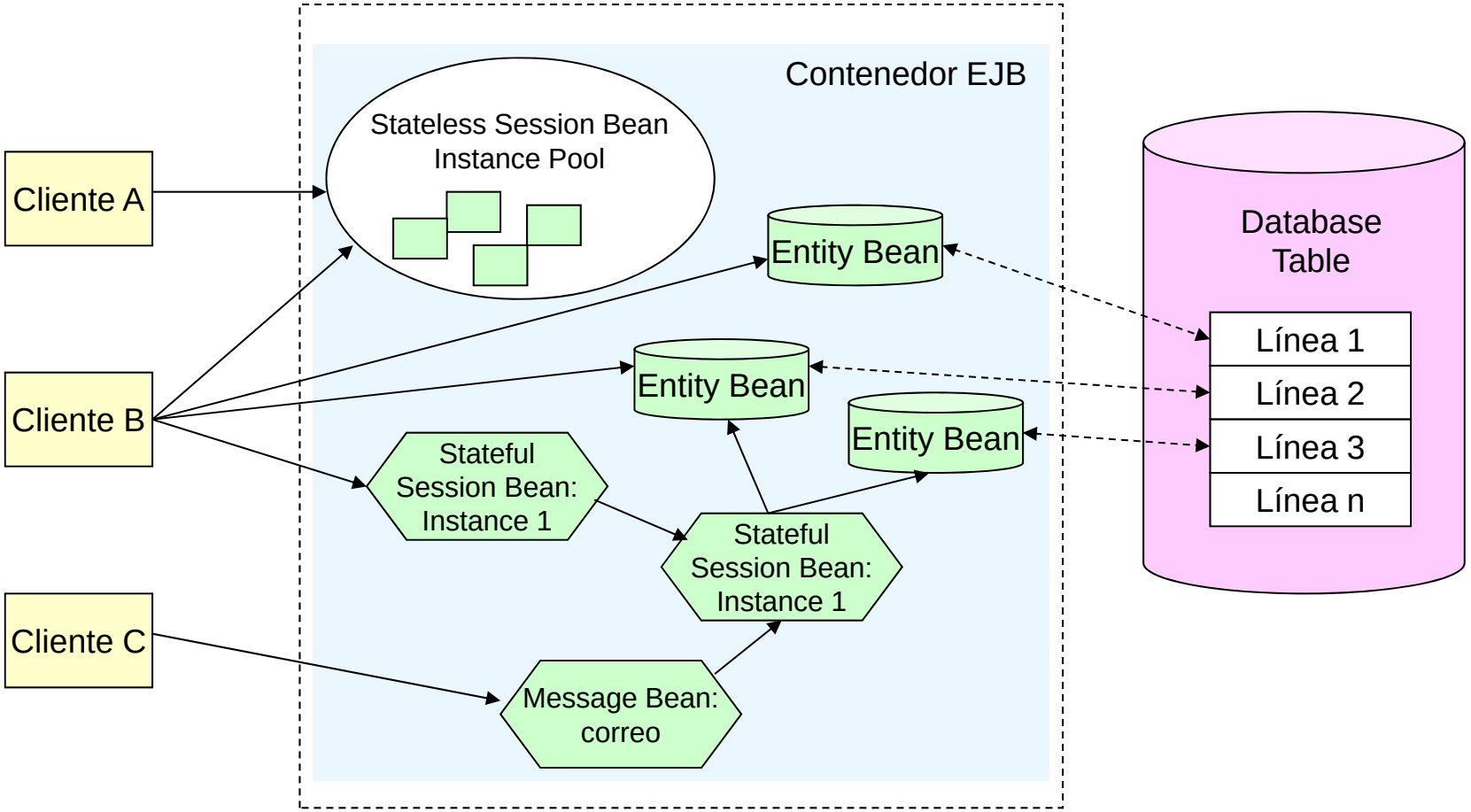
Entity Beans

- Modelan conceptos y objetos de negocio cuyos datos son persistentes
  - Típicamente la base de datos
- Pueden ser usados por varios clientes de forma conjunta y simultánea
  - Soporte transaccional
- Identidad visible externamente: *clave primaria*
- Larga duración (tanto como los datos asociados): incluso sobreviven caídas del ordenador y de la JVM
- Persistencia en Entity Beans:
  - **Gestionada por beans**
  - **Gestionada por el contenedor**

Message Beans

- Receptores de mensajes
- Intermediario entre cliente emisor y Message-driven Bean: servicio de mensajería
- Diferencia con Session y Entity Beans: comunicación asíncrona en lugar de llamada síncrona a métodos (el cliente se bloquea hasta el fin de la llamada)
- Emisor y Message-driven Beans son mutuamente anónimos
- No representan ningún tipo de objeto en la base de datos, aunque podrían accederla
- Los Message-driven Beans no poseen identidad
  - No pueden mantener información del estado del emisor, como los stateless Session Beans
- Los mensajes entrantes son capturados por el contenedor, quien los redirige a una instancia de Bean

Ejemplo de una aplicación compleja



Entity Bean vs. Stateful Session Bean (I)

Área funcional	Session Bean	Entity Bean
Estado objeto	Mantenido por contenedor en memoria principal entre transacciones. Swapp a almacenamiento secundario tras desactivación	Mantenido en BD. Típicamente caché en memoria en una transacción
Compartición objeto	Sólo puede ser usado por un cliente	Puede ser compartido por múltiples clientes. Un cliente puede pasar una referencia al objeto a otro cliente
Externalización de estado	El contenedor mantiene el estado del objeto internamente. Estado inaccesible a otros programas	Estado almacenado en BD. Otros programas (query SQL) pueden acceder al estado

Entity Bean vs. Stateful Session Bean (y II)

Área funcional	Session Bean	Entity Bean
Transacciones	El estado puede ser sincronizado mediante una transacción, pero no es recuperable	Estado cambiado de forma transaccional y es recuperable
Recuperación de fallos	No se garantiza que sobreviva un fallo y re-arranque del contenedor. Las referencias al objeto sesión pueden ser inválidas tras el fallo	Sobrevive el fallo y re-arranque del contenedor. El cliente puede usar las mismas referencias

Eligiendo entre Entity o Session Bean

- Las entidades de negocio se implementan típicamente como Entity Beans
- Los procesos de negocio conversacionales se implementan típicamente como Session Beans
- Los procesos de negocio colaborativos se implementan típicamente como Entity Beans
  - El estado representa los pasos intermedios realizados

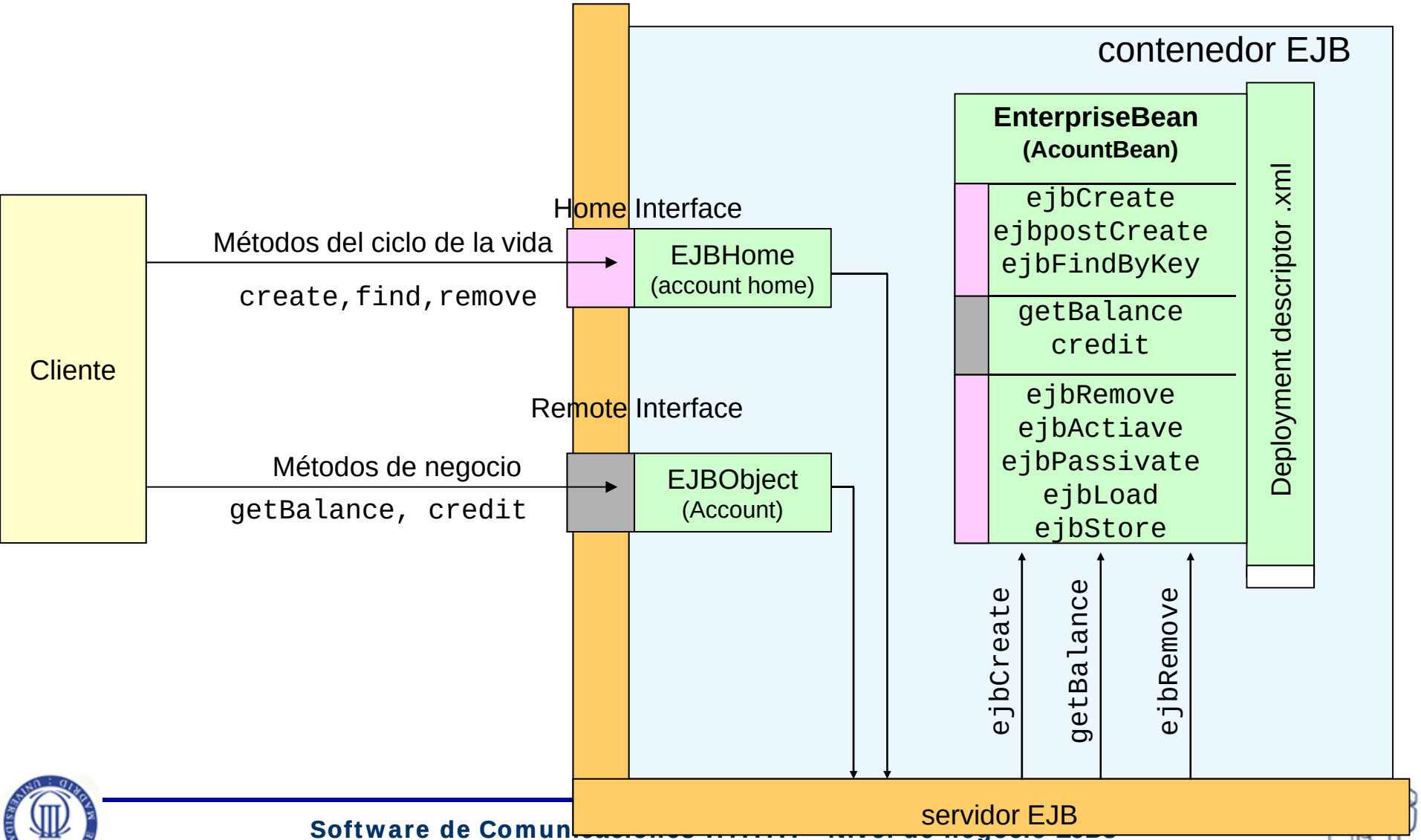
Cuándo utilizar Message Beans

- Cuando el cliente no ha de bloquearse en una operación arbitrariamente larga
  - Por ejemplo el envío de un correo electrónico confirmando una adquisición en Internet
- Con un sesion bean el cliente se podría bloquear innecesariamente en algunos casos
  - La página web no devolvería los resultados hasta que el correo no hubiese llegado al destino

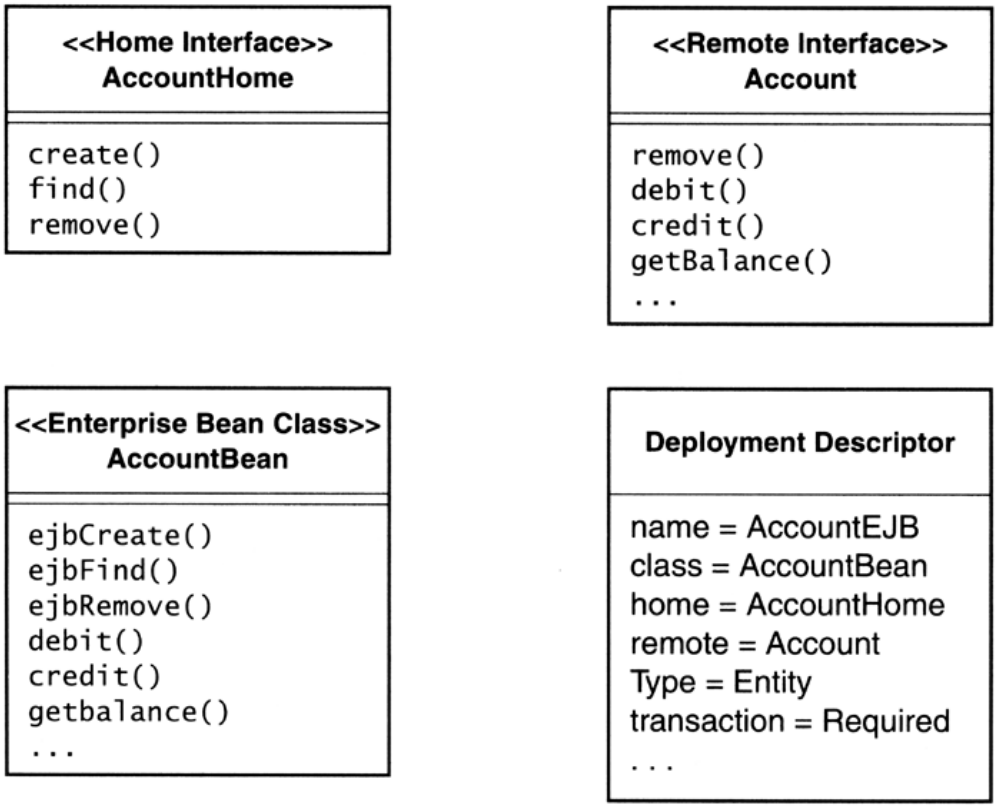
Estructura de un Enterprise Java Bean

- Consta de tres partes fundamentales:
  - **Clase Enterprise Bean**:
    - Métodos de negocio (definido por la aplicación)
    - Métodos del ciclo de vida (llamados por el contenedor)
  - **Interfaz de cliente**:
    - Home Interface (`create`, `remove` y `find`)
    - Remote Interface (métodos de negocio como `getSaldo`)
  - **Descriptor de despliegue**:
    - Información (xml) sobre el EJB y su entorno
      - Nombre del EJB
      - Nombre de los interface Home y Remote
      - Nombre de la clase EJB
      - Tipo del EJB
      - Servicios que el EJB espera de su contenedor
      - Entradas del entorno EJB (dependencias con otros EJBs y gestores de recursos)

Estructura de un Enterprise Java Bean



Un ejemplo: Una cuenta bancaria



Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

Detalles de la interfaz Home

```
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import java.util.Collection;
public interface AccountHome extends javax.ejb.EJBHome{
     // create methods
     Account create(string lastName, String firstName)
             throws RemoteException,
             CreateException, BadNameException;
     Account create(string lastName)throws
                     RemoteException, CreateException;
     // find methods
     Account findByPrimaryKey(AccountKey primaryKey)
             throws RemoteException, FinderException;
     Collection findInActive(Date sinceWhen) throws
             RemoteException, FinderException,
                     BadDateException;
  }//@AccountHome
```

Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

Detalles de EJBHome Interface

```
import java.rmi.RemoteException;
public interface EJBHome extends java.rmi.Remote{
     //Removes the enterprise bean
     void remove(Handle handle)
            throws RemoteException, RemoveException;
     void remove(Object primaryKey)
            throws RemoteException, RemoveException;
     //To interact with scripts
     EJBMetaData getEJBMetaData()
            throws RemoteException;
     //To get HomeHandle
     HomeHandle getHomeHandle()
            throws RemoteException;
}
```

Ejemplo de interfaz remota

```
import java.rmi.RemoteException;
 public interface Account extends javax.ejb.EJBObject {
        BigDecimal getBalance()
                                throws RemoteException;
        void credit(BigDecimal amount)
                                throws RemoteException;
        void debit(BigDecimal amount)
                                throws RemoteException,
                                InsufficientFundsException;
   }
```

La interfaz EJBObject

```
import java.rmi.RemoteException;
public interface EJBObject extends java.rmi.Remote{
   public EJBHome getEJBHome()
                     throws RemoteException;
   public Object getPrimaryKey()
                     throws RemoteException;
   public void remove()
               throws RemoteException, RemoveException;
   public Handle getHandle()
               throws RemoteException;
   boolean isIdentical(EJBObject obj2)
               throws RemoteException;
}
```

Detalles de la clase AccountBean

```
import java.rmi.RemoteException;
public class AccountBean implements javax.ejb.EntityBean{
   // life cycle methods from home interface
   public AccountKey ejbCreate(String lastName,String
                firstName)throws CreateException,
                          BadNameException {...};
   public AccountKey ejbCreate(String lastName)
                          throws CreateException {...}
   public void ejbPostCreate(String lastName, firstName)
                          throws CreateException,
                BadNameException {...};
   public void ejbPostCreate(String lastName)
                          throws CreateException {...}
   public AccountKey ejbFindByPrimaryKey(AccountKey primaryKey)
                          throws FinderException {...}
   public Collection ejbFindlnActive(Date sinceWhen)
        throws FinderException, BadDateException {...}
   // business methods from remote interface
   public BigDecimal getBalance() {...}
   public void credit(BigDecimal amount){...}
   public void debit(BigDecimal amount)
        throws InsufficientFundsException{...}
```

Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

Detalles de la clase AccountBean (cont.)

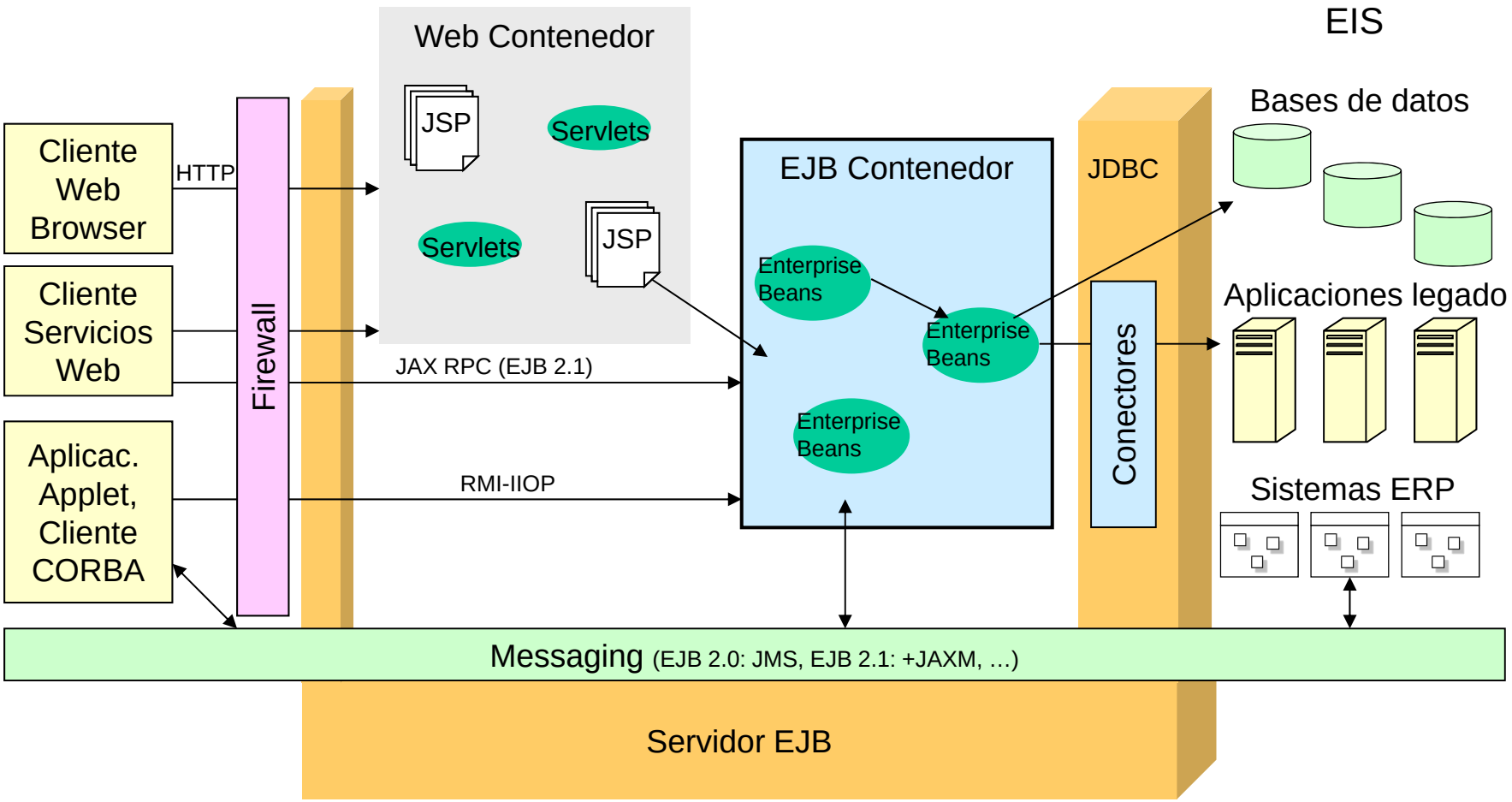
```
   ...
   // container callbacks from EntityBean interface
   public void ejbRemove() throws RemoveException {...}
   public void setEntityContext(EntityContext ec) {...}
   public void unsetEntityContext(EntityContext ec){...}
   public void ejbActivate(){...}
   public void ejbPassivate(){...}
   public void ejbLoad(){...}
   public void ejbStore(){...}
}
```

Descriptor de descripción

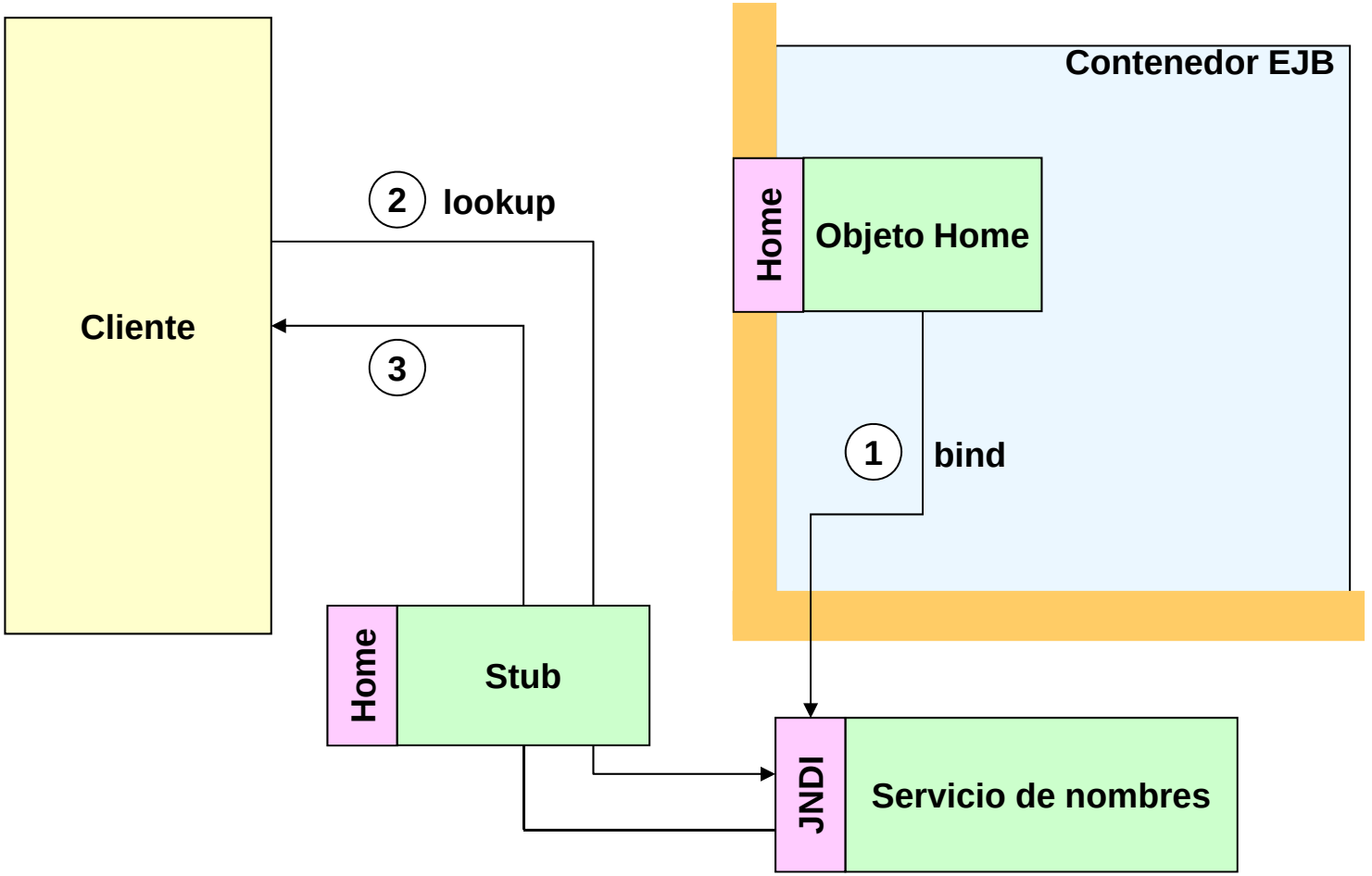
```
<entity-bean>
   <ejb-name>AccountEJB</ejb-name>
   <home>com.wombat.AccountHome</home>
   <remote>com.wombat.Account</remote>
   <ejb-class>com.wombat.AccountBean</ejb-class>
   <persistence-type>Bean</persistence-type>
   <prim-key-class>com.wombat.AccountKey</prim-key-class>
   ...
</entity-bean>
...
<container-transaction>
   <method>
        <ejb-name>AccountEJB</ejb-name>
        <method-name>*</method-name>
   </method>
   <trans-attribute>Required</trans-attribute>
</container-transaction>
...
```

Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

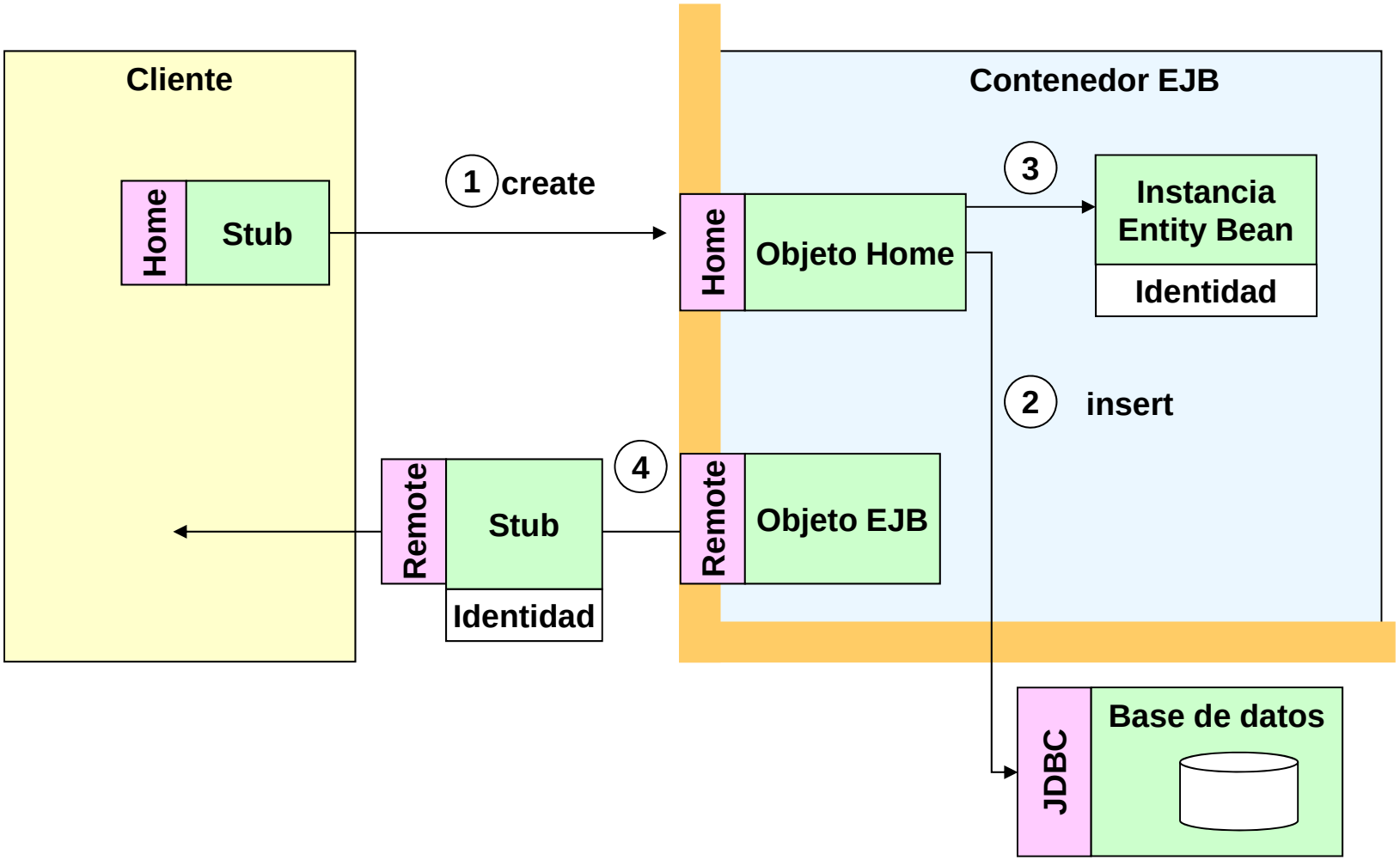
Comunicación con el EJB



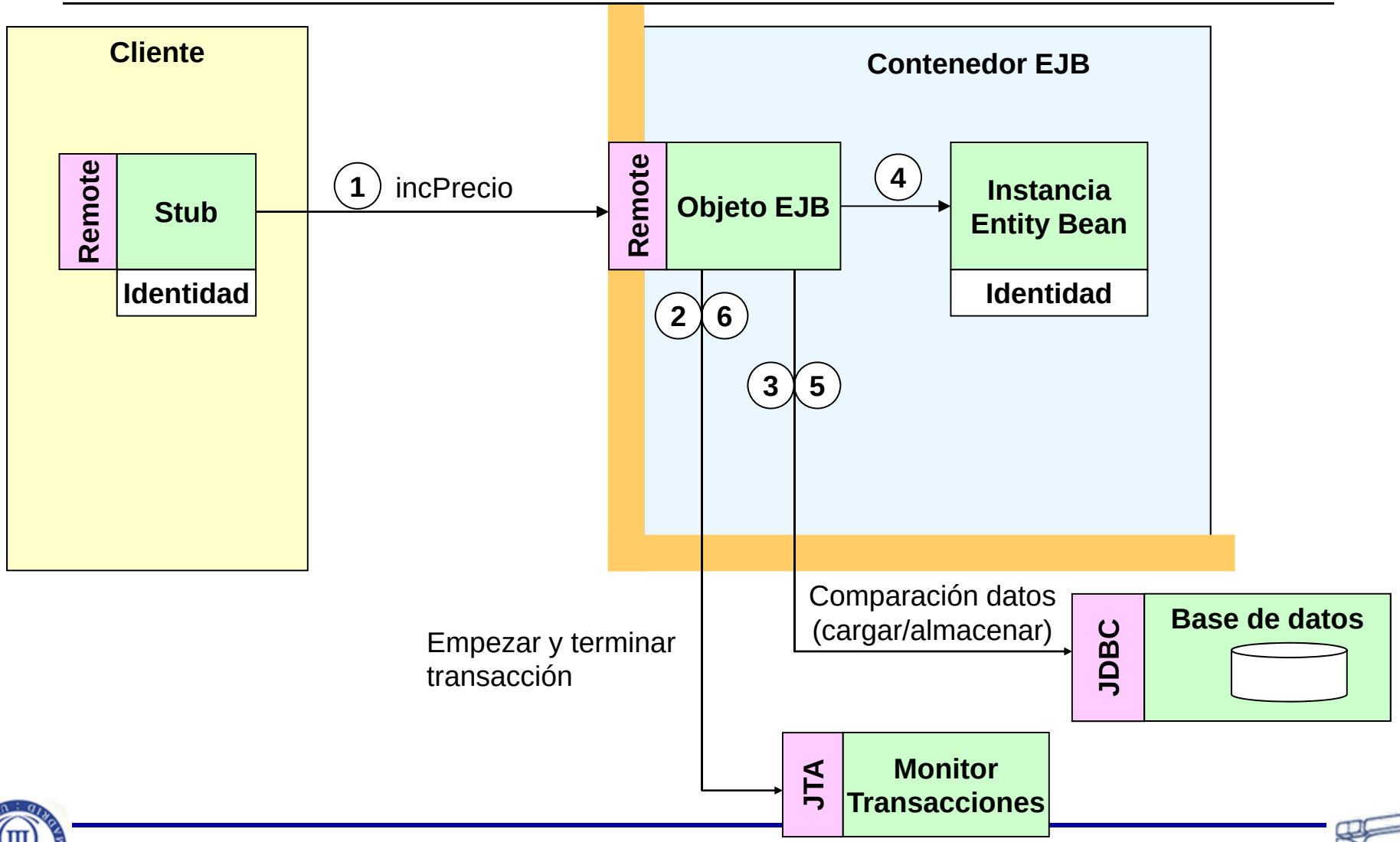
Accediendo a la referencia del EJB



Creando de un Entity Bean



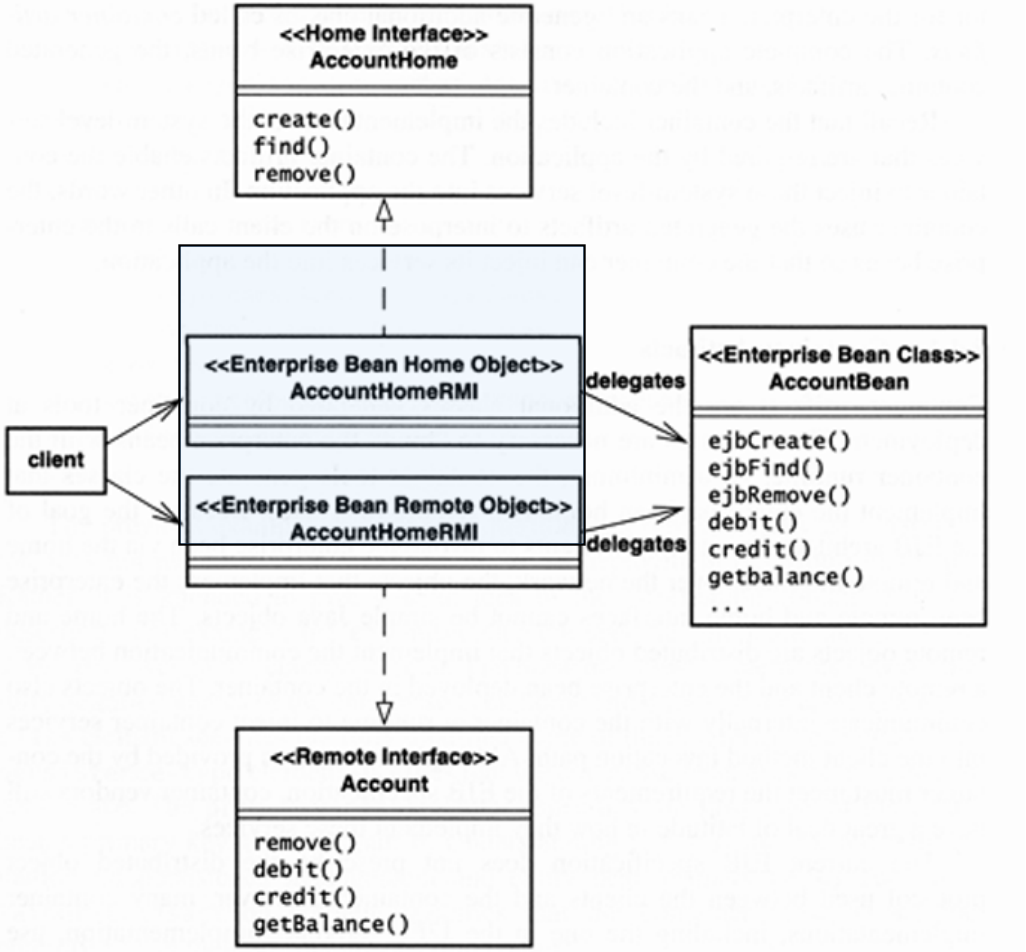
Invocando un método de un EJB



Herramientas del contenedor

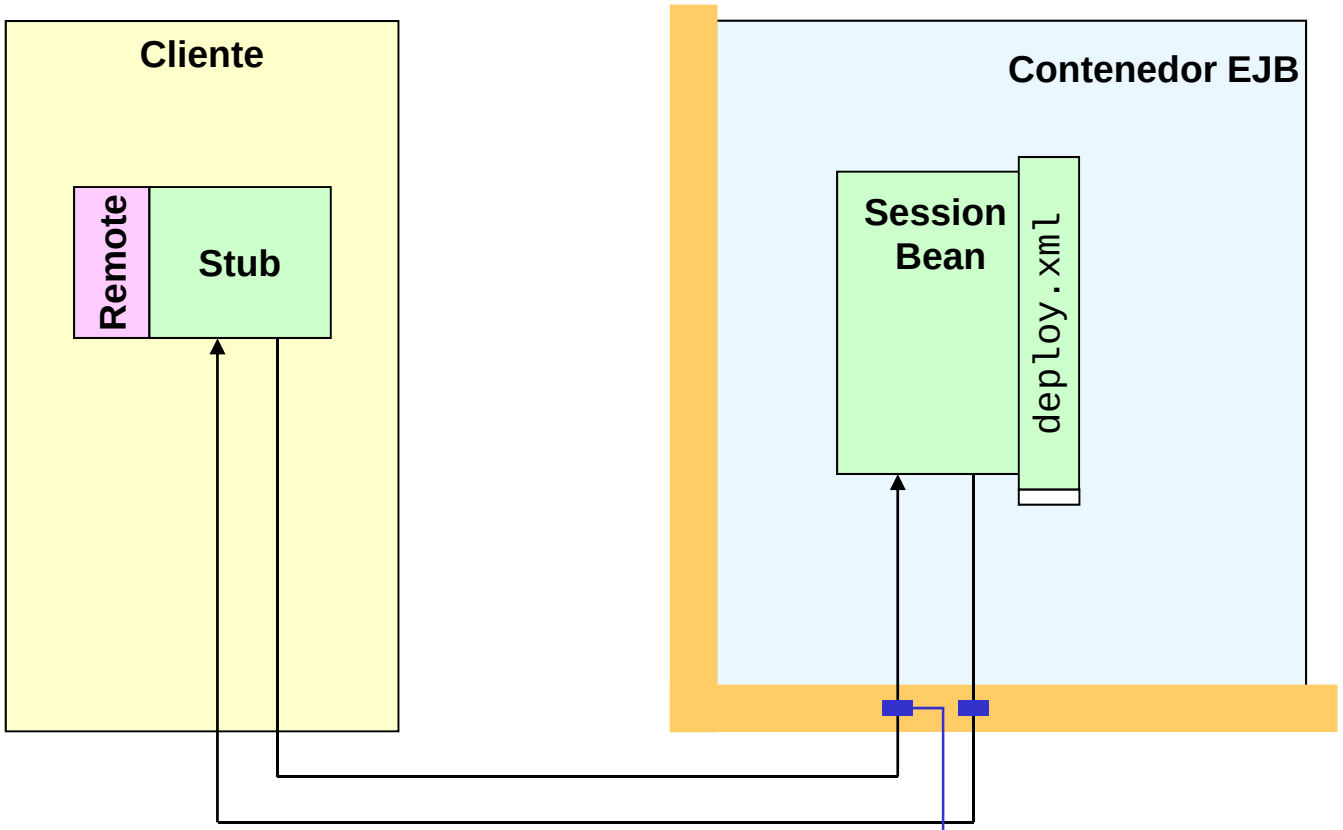
- Elementos de una aplicación operativa:
  - EJBs (lógica de negocio)
  - Contenedor (implementación de los servicios de nivel de sistema)
  - Artefactos de contenedor:
    - Las herramientas del contenedor leen el deployment descriptor y generan clases adicionales llamadas artefactos de contenedor (container *artifacts*)
- Los artefactos de contenedor permiten al contenedor inyectar los servicios de nivel de sistema (intercepción)

Artefactos del contenedor (despliegue)



Fuente:
Applying Enterprise JavaBeans™:Component-Based Development for the J2EE™ Platform, Second Edition
By: Vlada Matena; Sanjeev Krishnan; Linda DeMichiel; Beth Stearns **Publisher:** Prentice Hall

Proceso de intercepción e inyección de servicios



Transacciones
Seguridad
Distribución
Balance de carga
Concurrencia
Persistencia
Recuperación
...

```
if !autorized(){
return SecurityException()
}
```

Servicios ofertados por el contenedor al EJB

- **Administración de instancias**:
  - Gestión del ciclo de vida de las instancias
  - Los estados y procesos del ciclo de vida dependen del tipo de Bean (stateful vs. stateless)
- **Acceso remoto**:
  - El servidor EJB proporciona protocolos de comunicación para el acceso remoto a objetos distribuidos (RMI-IIOP)
  - El interfaz y semántica de llamada deben seguir las convenciones de Java RMI
  - El protocolo de comunicación debe soportar IIOP (según especificación CORBA)
  - La gestión de la distribución de las llamadas remotas es tarea del contenedor

Servicios ofertados por el contenedor al EJB (II)

- **Seguridad**:
  - Autorización de acceso a componentes: concepto declarativo basado en roles: En el Deployement Descriptor (descriptor de instalación) se definen roles de usuario y sus derechos de acceso a los métodos de los componentes
  - El instalador (deployer) asigna roles a los usuarios, la gestión de usuarios y roles la realiza el servidor EJB, el control de acceso el contenedor
- **Persistencia**:
  - Las instancias de Entity Beans en la memoria de trabajo pueden estar enlazadas con datos de negocio de cualquier EIS. El contenedor garantiza la consistencia de datos (carga y almacenamiento periódicos)
    - Persistencia gestionada por Beans: la instancia usa las conexiones JDBC a bases de datos proporcionadas por el contendor
    - Persistencia gestionada por contenedor: en general se soportan bases de datos relacionales
    - El medio de persistencia depende del contenedor y es independiente del Bean

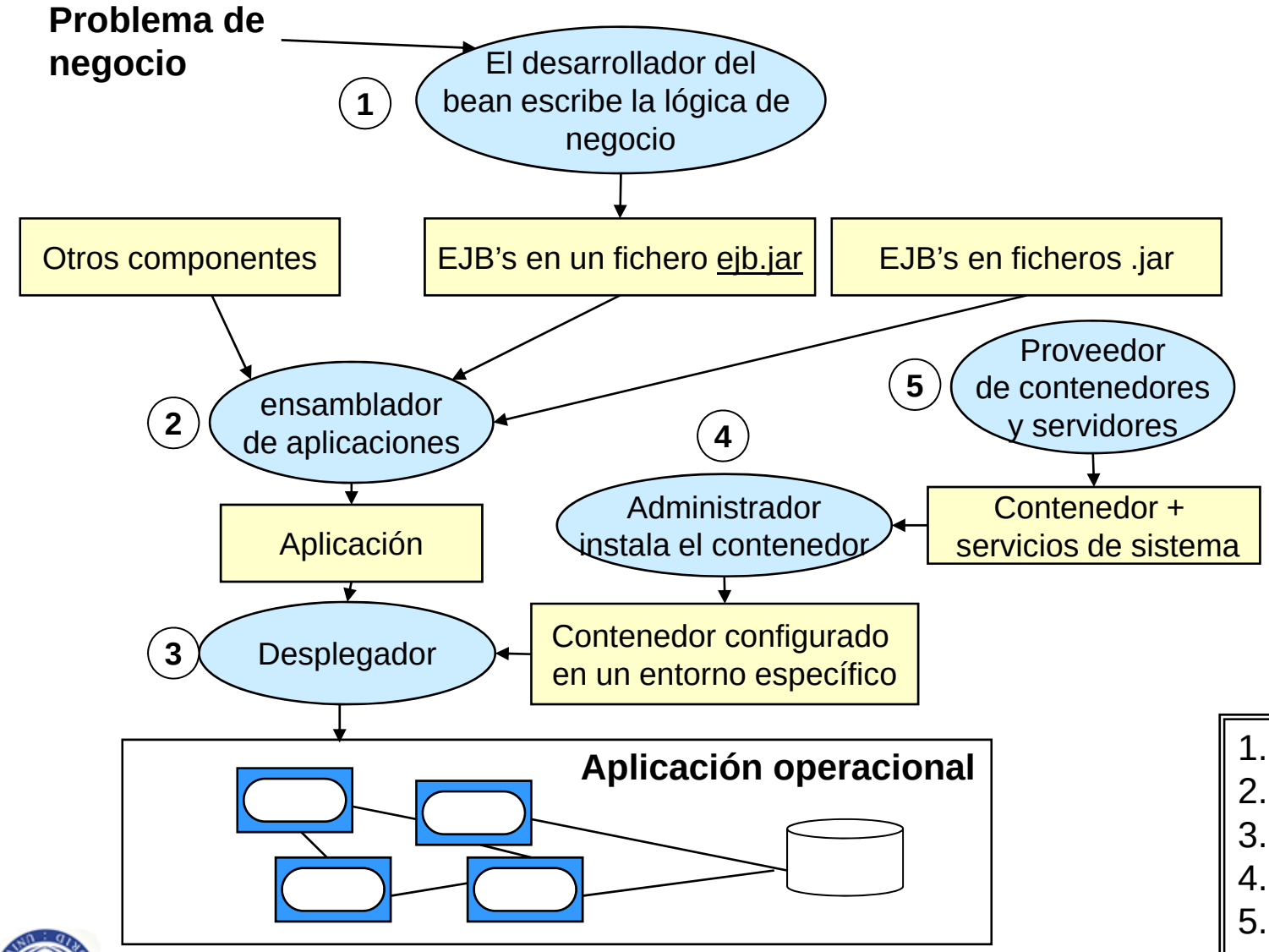
Servicios ofertados por el contenedor al EJB (III)

- **Transacciones**:
  - Secuencia de acciones (accesos a datos) ejecutadas de forma “atómica” (no interrumpida), evitando problemas por acceso concurrente a datos, que se puede “deshacer” por completo en caso de fallo
  - Coordinación de transacciones mediante un monitor de transacciones
  - El contenedor proporciona los protocolos para manejo de transacciones (por ejemplo 2-Phases-Commit-Protocol)
  - El contenedor proporciona al Bean una interfaz única JTA (Java Transaction API)
  - Las transacciones pueden ser empezadas y terminadas por el Bean o dejadas al control del contenedor (especificando en el Deployment Descriptor qué métodos deben ser protegidos por transacciones)

Servicios ofertados por el contenedor al EJB (IV)

- **Servicio de nombres y directorios**:
  - Asociación de referencias a objetos con nombres en una estructura de directorios jerárquica (JNDI API: Java Naming and Directory Interface)
  - El contenedor asocia los Beans de forma automática, y proporciona además diversa información a los Beans en un directorio privado (Entorno)
- **Mensajería**:
  - El contendor proporciona a los beans acceso al servicio de mensajería mediante el API JMS (Java Messaging Service): comunicación asíncrona de mensajes entre dos o más participante mediante un sistema de colas de mensajes
  - Los receptores de mensajes son los “Message-Driven Beans” y cualquier Enterprise Bean puede ser emisor

Diferentes roles en los EJB



Herramientas

- El desarrollo de EJB's requiere habitualmente la utilización de herramientas auxiliares:
  - **IDE** para el desarrollo de EJB's y sus diferentes interfaces
  - **Data Access** para configurar la base datos y acceder a sus recursos (borrar datos, crear tablas, ..)
  - **UML** para facilitar el diseño de la lógica de la aplicación empresarial
  - **Edición Web** para facilitar el diseño de páginas web complejas (flash, …)
  - **De Integración** para interactuar EIS preexistentes como por ejemplo sistemas ERP
  - **Despliegue** para resolver dependencias entre diferentes EJB's y el entorno operacional
  - **Gestión del contenedor y del servidor** para monitorizar y controlar el estado del servidor

Cuestiones para reflexionar

Cuestiones para reflexionar – Bloque I

- Un EJB puede, por si solo, constituirse en una aplicación empresarial J2EE o precisa, por el contrario, de algún otro tipo de componente adicional.

Cuestiones para reflexionar – Bloque II

- Indique qué tres tipos de EJB define la especificación J2EE, mencionando además el tipo de funcionalidad que oferta cada uno de ellos.
- Existe algún caso en el que un proceso de negocio se almacene en un Entity Bean. Justifique su respuesta.

Cuestiones para reflexionar – Bloque III

- Que tres partes fundamentales forman parte de un Enterprise Java Bean. Enumérelas indicando que información almacena cada una de ellas.

Cuestiones para reflexionar – Bloque IV

- Qué es un artefacto del contenedor y para que sirve.
- Indique cuales son los siete servicios que le son ofertados a un EJB

Cuestiones para reflexionar – Bloque V

- De todos los roles existentes en los EJB, indique aquellos que piensa que utilizará en el laboratorio