

TITULO

# APLICACIONES EMPRESARIALES II

**Semana Nro 01**

Arquitectura de  
**Aplicaciones Empresariales Java EE**

# Aplicaciones Empresariales II

## Esquema de Notas

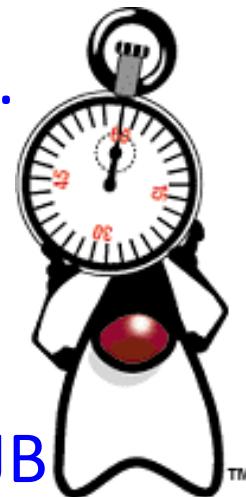
|                    |              |
|--------------------|--------------|
| Ex. Parcial        | → 20%        |
| Ex. Final          | → 30%        |
| Pr. Pract.         | → 20%        |
| <b>Proy. Final</b> | <b>→ 30%</b> |

### **Proyecto Final (30%)**

|   |       |
|---|-------|
| Avance 1                                  | → 20% |
| Avance 2                                  | → 20% |
| Avance 3                                  | → 20% |
| Av. Final y Sustentación                  | → 40% |
| Si no exponen la nota de proyecto es CERO |       |

# Agenda

- El servidor de aplicaciones.
- Arquitectura del Servidor de aplicaciones Java EE.
- Generación de pool de conexiones.
- Generación de un data source.
- Las características de los Enterprise JavaBeans
- Los diferentes componentes de la arquitectura EJB
- Roles y responsabilidades provistas por la especificación EJB
- Distintos tipos de los Enterprise JavaBeans
- Componentes de los Enterprise JavaBeans

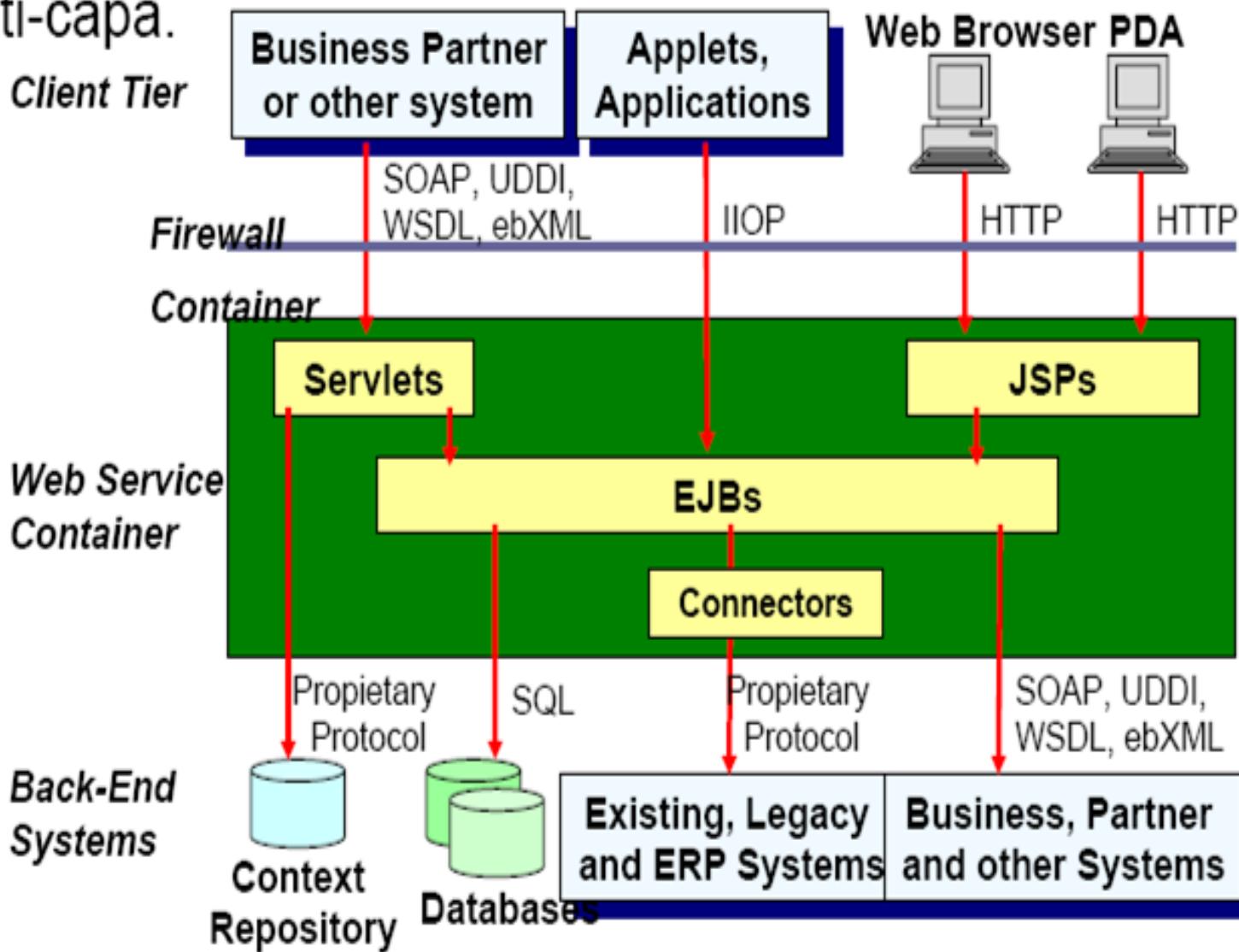


# **Arquitectura de Aplicaciones Empresariales**



# Servidores de aplicaciones Java EE

El modelo de aplicaciones J2EE se basa en una arquitectura multi-capa.



# Servidores de Aplicaciones J2EE

## Implementaciones comerciales y gratuitas de J2EE

- Implementaciones J2EE:

- IBM WebSphere
  - JRun Server (Macromedia)
  - BEA WebLogic
  - Borland Enterprise Server
  - IONA iPortal
  - JBOSS (Free)
  - Oracle 9iAS

# Arquitectura de los EJBs

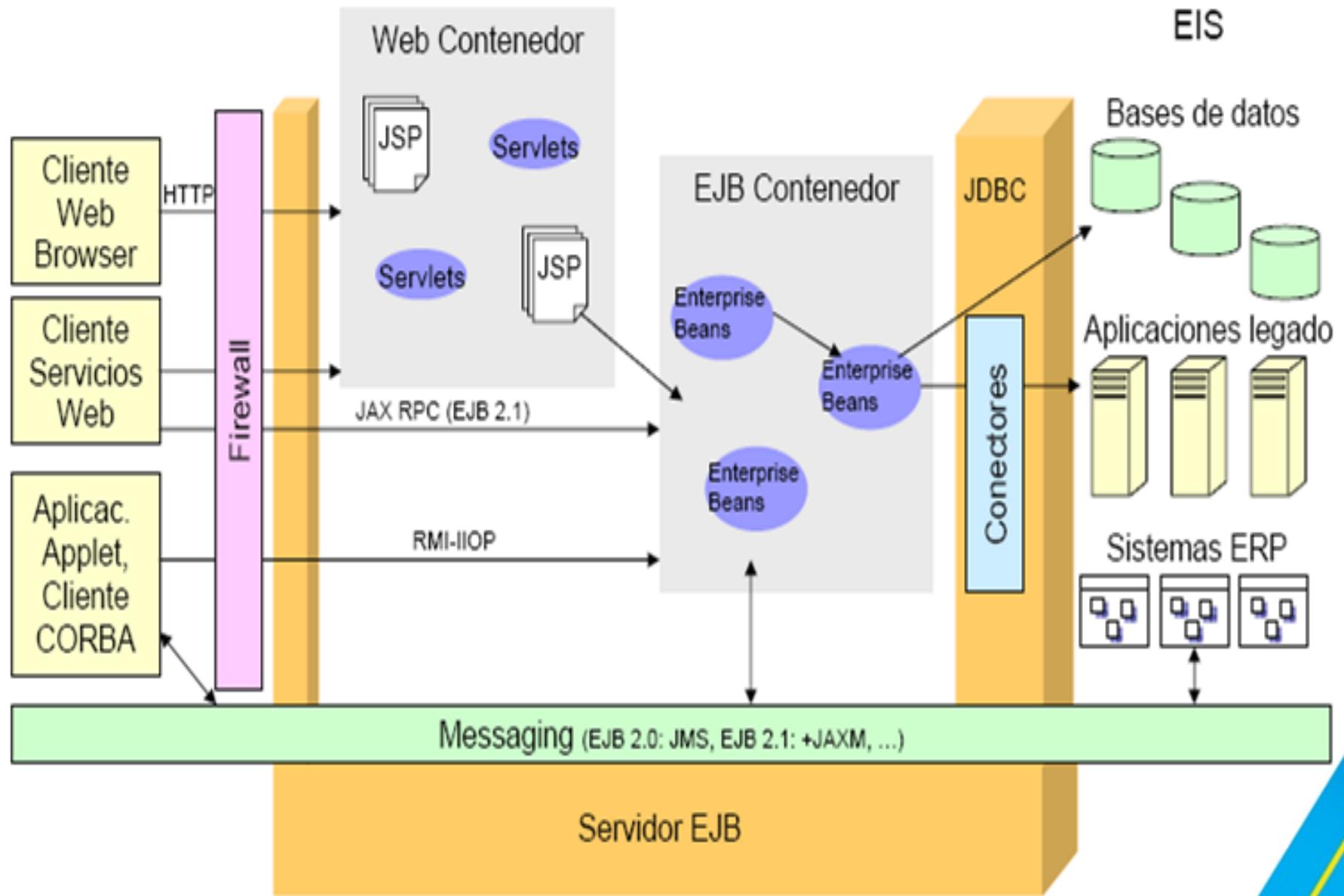
## Qué son los Enterprise JavaBeans?

- EJBs: componentes usados como parte de aplicaciones corporativas distribuidas
- Cada bean encapsula parte de la lógica de negocio de la aplicación
- Se comunica con gestores de recursos, y con otros EJBs
- Accedido por distintos tipos de clientes: EJBs, servlets, clientes de aplicación, etc.
- En tiempo de ejecución, reside en un contenedor EJB: servicios de seguridad, transacción, instalación (deployment), concurrencia, y gestión del ciclo de vida
- Una aplicación puede tener uno o varios EJBs en uno o varios contenedores EJB

## Versiones

- EJB 1.x → ya muy antiguo
- **EJB 2.0 → Versión poco usada**
- EJB 2.1 → Cambios menores como soporte de servicios Web
- EJB 3.0 → Uso de anotaciones de Java 5 para simplificar el desarrollo

# ISIV / Servidores de Aplicaciones J2EE



# Contenedor EJB

## Facilidades del Contenedor

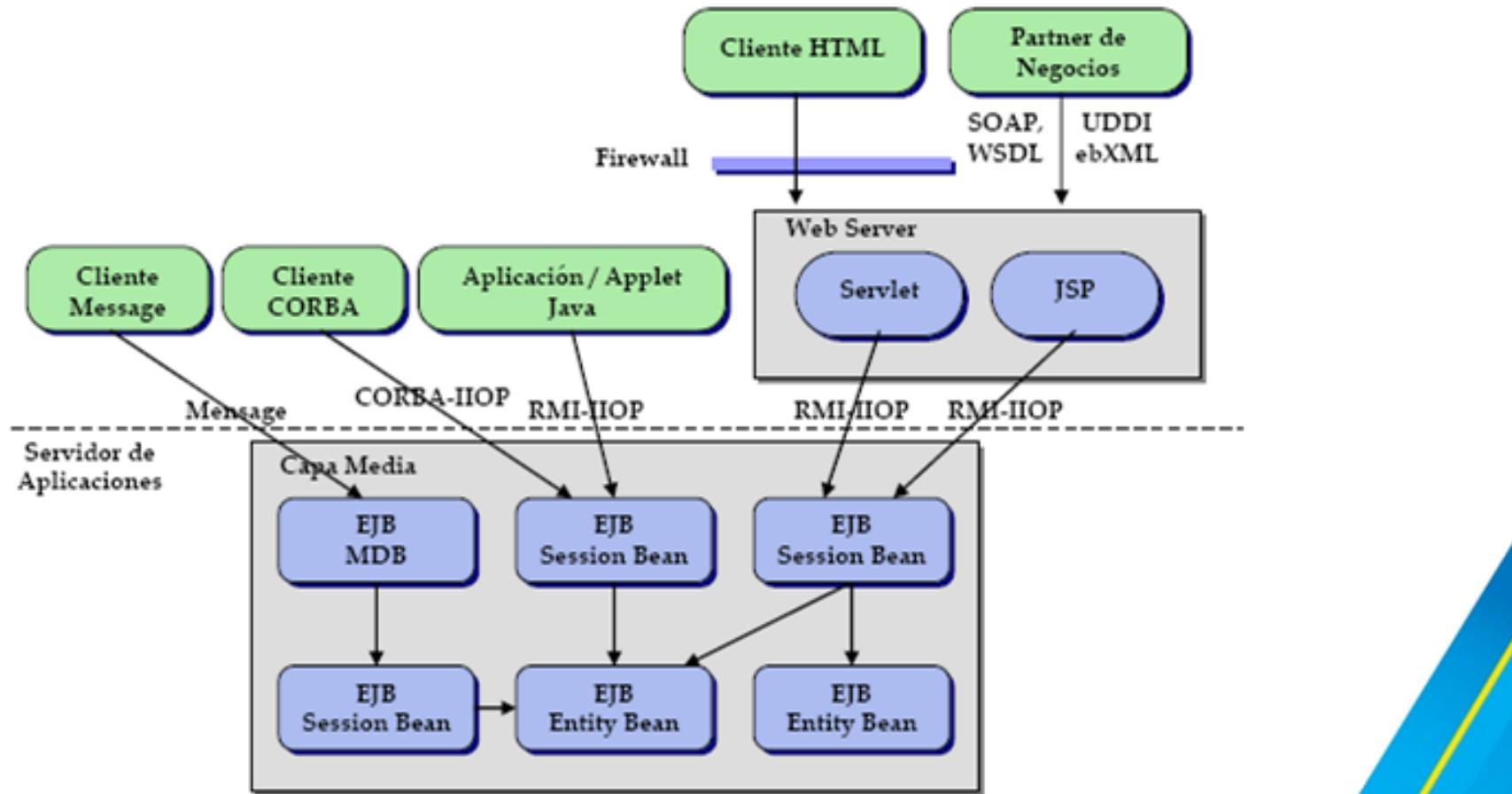
- El **Contrato con el Componente** especifica un conjunto de APIs que los componentes que residen en dicho container deben implementar. Dichas APIs definen el mecanismo de comunicación entre el contenedor y el componente.
- Las **APIs de Servicios** proveen acceso a las extensiones estándar definidas por J2EE
  - Ej: JDBC, JTA, JNDI, JMS, etc
- Los **Servicios Declarativos** permite especificar el uso de servicios o comportamientos de un componente declarativamente por fuera de la implementación del mismo. Ej. de estos servicios pueden ser: Manejo transaccional, seguridad, etc
- Los **Otros Servicios** que provee el contenedor son: manejo del ciclo de vida de un componente, pooling de recursos (ej. BDs), inicializar el servicio JNDI registrando los componentes EJB o servicios de Clustering.

## Tipos de EJBs

- **Session Beans:** **modelan procesos de negocios**, y pueden hacer cualquier cálculo necesario o acceder a otros sistemas.
- **Entity Bean:** **modelan la información del negocio.**
  - Persistentes a través de una base de datos,
  - cacheados en memoria durante una transacción,
  - pueden compartirse por múltiples clientes y un cliente puede pasar el objeto por referencia a otro.
  - El estado se cambia transaccionalmente y es recuperable.
    - Se recupera frente a fallas, y un cliente puede seguir utilizando la referencia al objeto luego que el container re-inicializa luego de la falla.
- **Message Driven Beans:** son “como” session beans pero **abocados a atender invocaciones por mensajes**. Están desde la versión 2.0 de EJB, y permiten incluir mensajería asincrónica en éste modelo.
- **Ejemplo:** combinado es utilizar un “Session Bean” para representar un retiro de un ATM, e invocar a un “Entity Bean” que representa la cuenta.

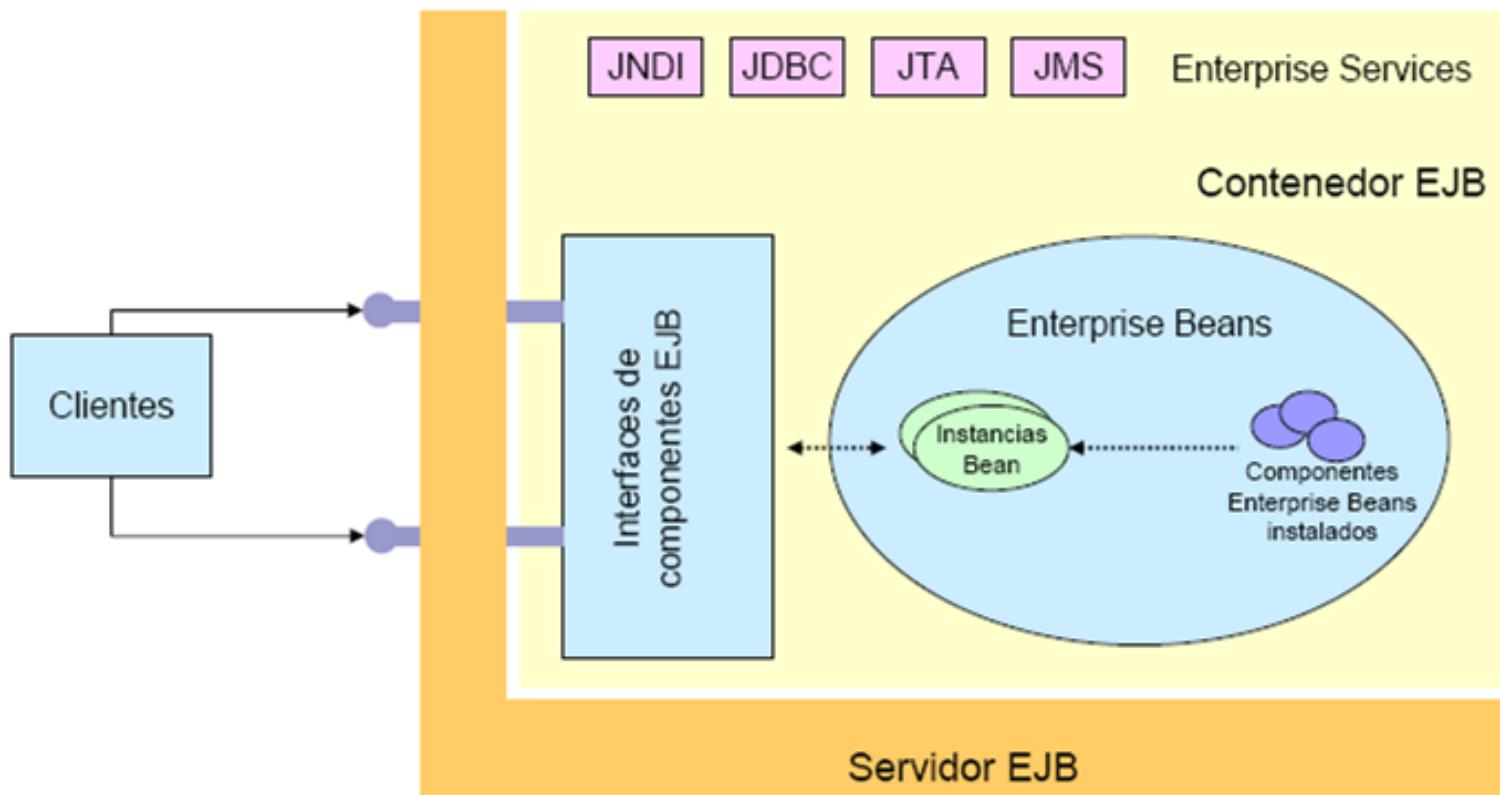
## Interacción con los componentes

- Tenemos invocaciones a “sessions” y MDBs pero no a “entities”.



# Arquitectura EJB

## Arquitectura EJB



# Arquitectura EJB

## Elementos de la arquitectura EJB

- **Servidor EJB:** gestión de conexiones, errores, disponibilidad, balance de carga y escalabilidad. Conexión con la estructura de comunicación y vigilancia de los procesos y threads que corren en el container
- **Contenedor EJB:** ofrece un entorno de ejecución a los componentes instalados: manejo de las instancias-Bean, control del ciclo de vida y acceso a los servicios Enterprise estándar  
El conjunto *Servidor EJB* y *Container EJB* proporciona un *servidor de aplicaciones*
- **Enterprise Beans:** son los componentes de servidor en los que está encapsulada la lógica de negocio. Hay tres tipos:
  - *Session Beans* representan procesos de negocio
  - *Entity Beans* representan datos de negocio
  - *Message-Driven Beans* procesan mensajes asíncronos
- **Clientes:** a EJB pueden acceder clientes basados en browser, clientes Java, Servlets, clientes CORBA u otros Enterprise Beans

# Capa del Negocio

## Entidades de negocio

- Objeto de negocio que representa información mantenida por la empresa
  - Estado, mantenido en la BD (persistente)
  - Ejemplos: cliente, pedido, cuenta, empleado
  - Las reglas de negocio asociadas con una entidad
    - Restringen los valores de su estado (Ej: código postal 5 cifras)
    - Mantienen relaciones entre entidades (Ej: un cliente con varios pedidos)

# Capa del Negocio

## Procesos de negocio

- Objeto de negocio que encapsula una interacción de un usuario con una entidad de negocio
  - Actualiza el estado de una entidad de negocio
  - Puede tener su propio estado
    - Estado persistente: proceso en varios pasos: *proceso de negocio colaborativo*
      - Ejemplo: procesamiento de una solicitud de préstamo
    - Estado transistorio: proceso de negocio completado por un actor durante una conversación: *proceso de negocio conversacional*
      - Ejemplo: sacar dinero de un cajero automático

# Capa del Negocio

## Reglas de negocio

- Distribuidas entre los componentes que implementan las entidades y procesos de negocio
  - En función de si la regla aplica a la entidad o al proceso
  - Ejemplo entidad: el balance de una cuenta no puede ser negativo (es independiente del proceso que lo cause)
  - Ejemplo proceso: no se pueden sacar más de 500€ de un cajero automático (independiente de la entidad Cuenta)

# Especificación EJB

## Enterprise Beans (recordemos)

- Componentes de servidor (especificación EJB), implementa la lógica de negocio
- Mantienen una identidad única durante todo su ciclo de vida
- 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. Transacciones bancarias, 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

# Especificación EJB

## 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
- 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

# Especificación EJB

## Entity Beans

- Modelan conceptos y objetos de negocio cuyos datos son persistentes
- Pueden ser usados por varios clientes de forma conjunta y simultánea -> transacciones
- Identidad visible externamente: *clave primaria*
- Larga duración (tanto como los datos asociados): incluso sobreviven caídas del ordenador
- Persistencia en Entity Beans: gestionada por beans o por contenedor, diferencia no visible para el cliente

# Especificación EJB

## Entity Bean vs. Stateful Session Bean

| 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 cliente. Un cliente puede pasar una referencia al objeto a otro cliente |
| Externalización de estado | El contenedor mantiene el estado del objeto internamente. Estado accesible a otros programas                            | Estado almacenado en BD. Otros programas (query SQL) pueden acceder al estado                              |

# Especificación EJB

## Entity Bean vs. Stateful Session Bean

| 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 rearrenque del contenedor. Las referencias al objeto sesión pueden ser inválidas tras el fallo | Sobrevive el fallo y rearrenque del contenedor. El cliente puede usar las mismas referencias |

# Especificación EJB

## Elección de Entity Bean 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 que requieran la acción de varios clientes se implementan típicamente como Entity Beans
  - El estado representa los pasos intermedios realizados

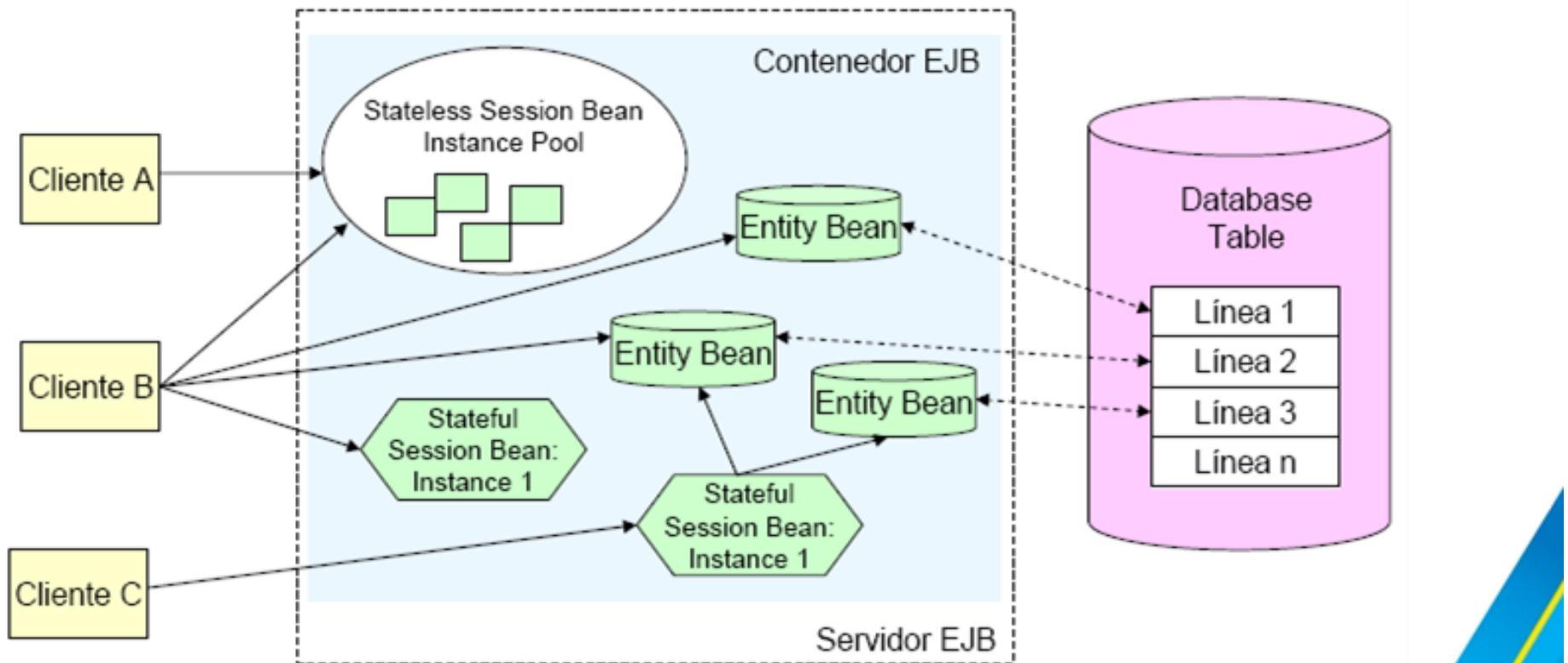
# Especificación EJB

## Message-driven 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
- 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

# Especificación EJB

## Ejemplo Enterprise Beans



# Especificación EJB

## Estructura de los Enterprise Beans

- **Clase Enterprise Bean:** implementa
  - métodos de negocio
  - métodos de ciclo de vida (llamados directamente por el contenedor)
- **API vista-cliente:** define
  - *Home interface*
    - métodos create, remove y find
  - *Remote interface*
    - Métodos de negocio

# Especificación EJB

## Estructura de los Enterprise Beans

- **Deployment descriptor:** Documento XML declara:
  - Información de los 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, como transacciones
    - Entradas del entorno EJB (dependencias con otros EJBs y gestores de recursos)

# Especificación EJB

## Vista cliente sobre Session y Entity Beans

- Los session y entity beans ofrecen distintos interfaces a los clientes
- Los message-driven beans no tienen clientes propiamente dichos sino que reaccionan a mensajes, por lo que no ofrecen interfaces sino que anuncian su canal de mensajes (JMS-Topic o JMS-Queue)
  - Distinto modo de programación y comportamiento que los session y entity beans
- Importante para los clientes:
  - Servicio de nombres y directorios a través de JNDI para localización de EJBs
  - Protocolo RMI-IIOP para llamadas remotas a métodos

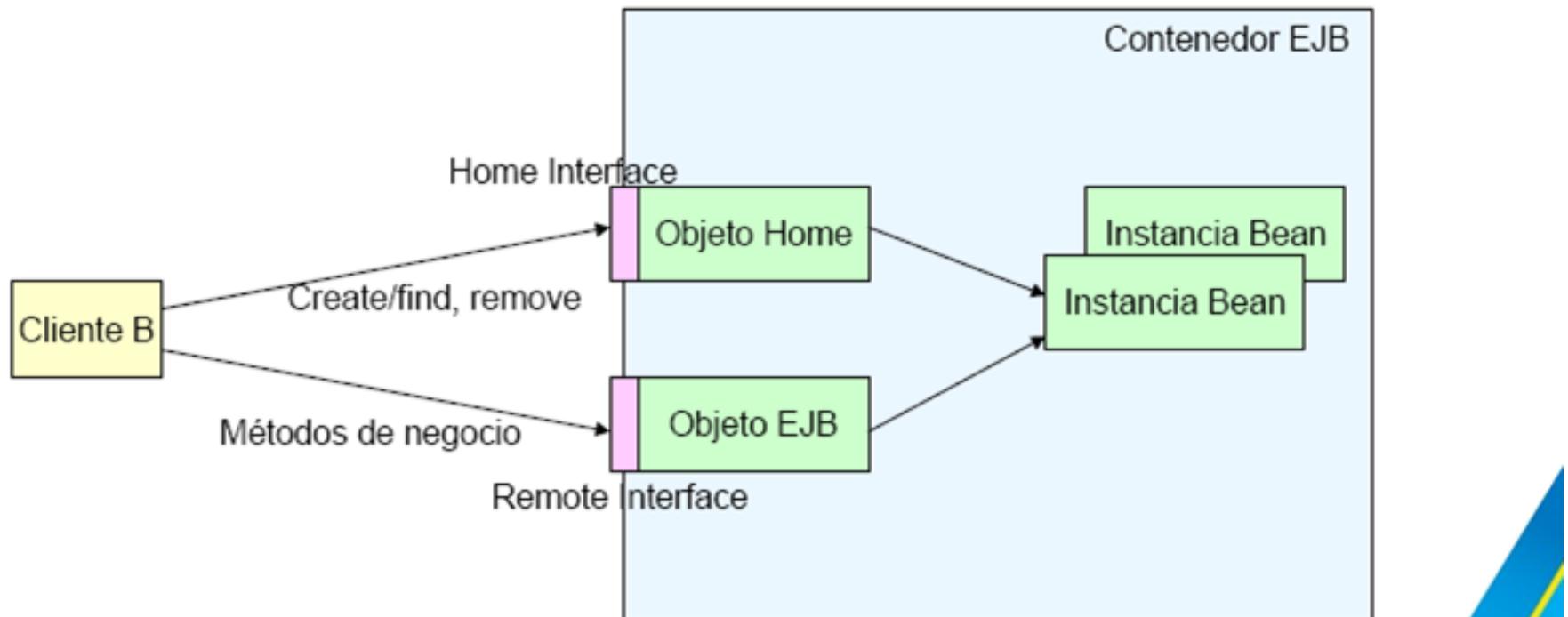
# Especificación EJB

## Características de la vista del cliente

- *Home Interface*: Crear y borrar beans (session y entity beans), encontrar (entity beans)
- *Component Interface*: Exporta los métodos de negocio
- Estos interfaces son implementados por Objetos-Home y Objetos-EJB: vista del cliente sobre los session y entity beans
  - Interfaces creados por el diseñador
  - Durante la instalación, el contenedor genera las clases que implementan los interfaces
- *Intercepción*: el cliente nunca opera directamente sobre las instancias de beans sino sobre objetos generados por el contenedor
  - El contenedor controla las llamadas y realiza pre- y post-procesado
  - Proporciona el entorno de ejecución a los EJBs llamados
    - Ej: Control de permisos, enlace dinámico del objeto-bean a las instancias-bean, generación de instancias-bean si es necesario (*Just-in-Time-Activation*), o creación previa y gestión (*Pooling*)

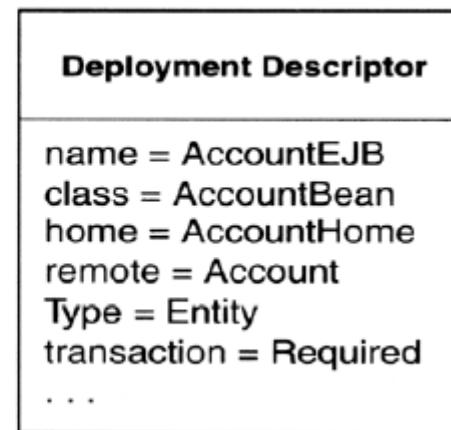
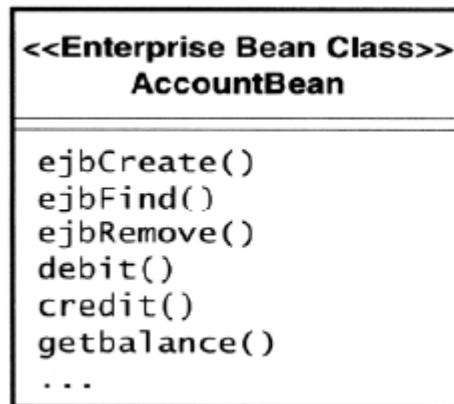
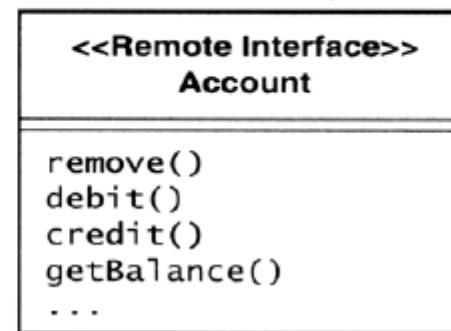
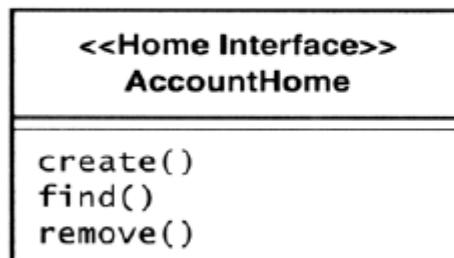
# Especificación EJB

## Esquema vista de cliente



# Especificación EJB

## Ejemplo: Account EJB



# Especificación EJB

## Características especiales de los Message Driven Beans

- Los MDB están suscritos a una cola o tópico
  - Cuando llega un mensaje el contenedor despierta al MDB para que procese el mensaje
- Algunas características de los MDB:
  - No tienen ni home, remote, local o local home interface
    - No es posible invocarlo directamente
    - Se invoca enviando un mensaje JMS
  - Solamente tienen un método de negocio
    - Es el invocado por el contenedor cuando llega un mensaje
    - El método se llama onMessage() que recibe un mensaje
  - No retornan un valor al cliente
  - No pueden levantar excepciones al cliente

# Especificación EJB

## Contenedor EJB

Entorno de ejecución para los componentes EJB instalados

### ■ Administración de instancias:

- Gestión del ciclo de vida de las instancias
- Los estados y procesos del ciclo de vida dependen del tipo de Bean

### ■ 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

# Especificación EJB

## Contenedor EJB (cont.)

### ■ Seguridad:

- Autorización de acceso a componentes: concepto declarativo basado en roles: En el Deployment 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 contenedor
  - Persistencia gestionada por contenedor: en general se soportan bases de datos relacionales
  - El medio de persistencia depende del contenedor y es independiente del Bean

# Especificación EJB

## Contenedor EJB (cont.)

### ■ 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 contendor 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)

# Especificación EJB

## Contenedor EJB (cont.)

### ■ Servicio de nombres y directorios:

- Asociación de referencias a objetos con nombres en una estructura de directorios jerárquica (JNDI API: Java Naminig 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 doso más participante mediante un sistema de colas de mensajes
- Los receptores de mensajes son los “Message-Driven Beans”, cualquier Enterprise Bean puede ser emisor

# Contenedor EJB

## Servicios contenedor mediante APIs

- Servicio de nombres - JNDI
- Servicios de despliegue (deployment)
  - Deployment Descriptor (XML files)
  - Deployment Units (EAR files)
- Servicios de transacción - JTA
- Servicios de seguridad - JAAS
- Java Database Connectivity - JDBC
- JavaMail/JAF (Java Application Framework)
- Java Messaging Services – JMS
- Java Api para procesamiento XML - JAXP
- Java Connector Architecture - JCA

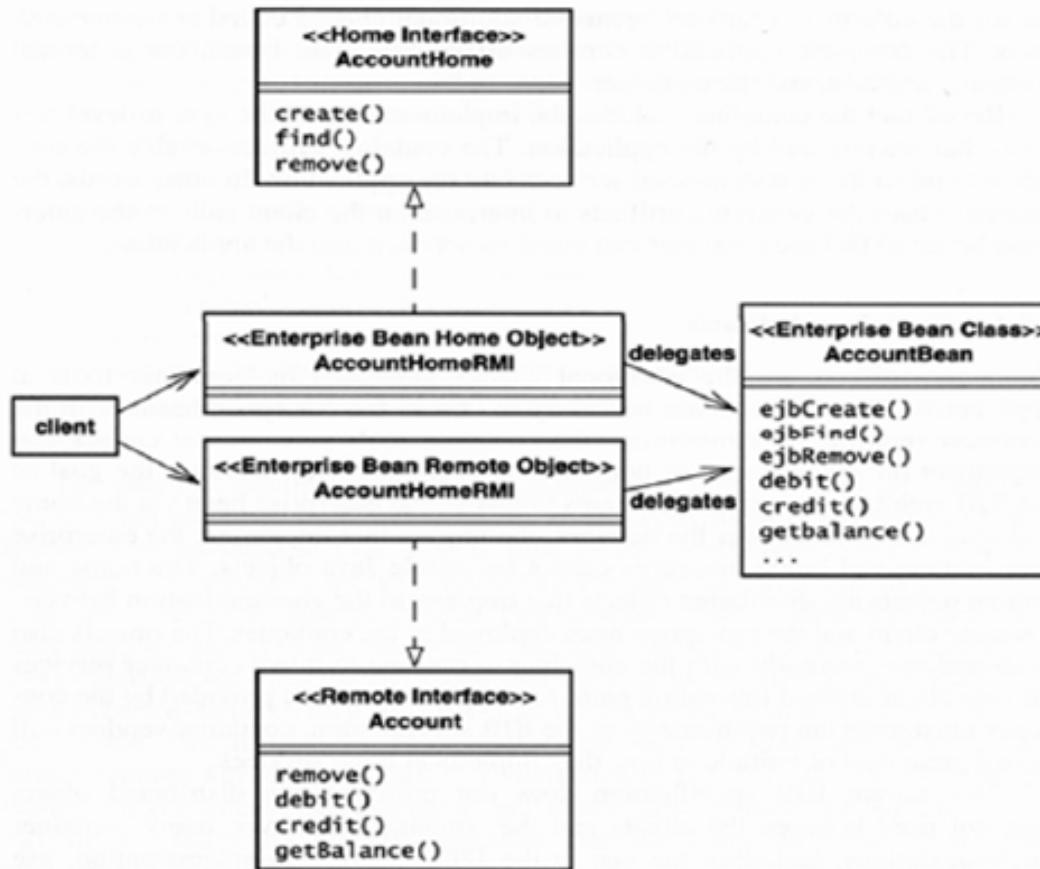
# Contenedor EJB

## Herramientas de 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 injectar los servicios de nivel de sistema (intercepción)

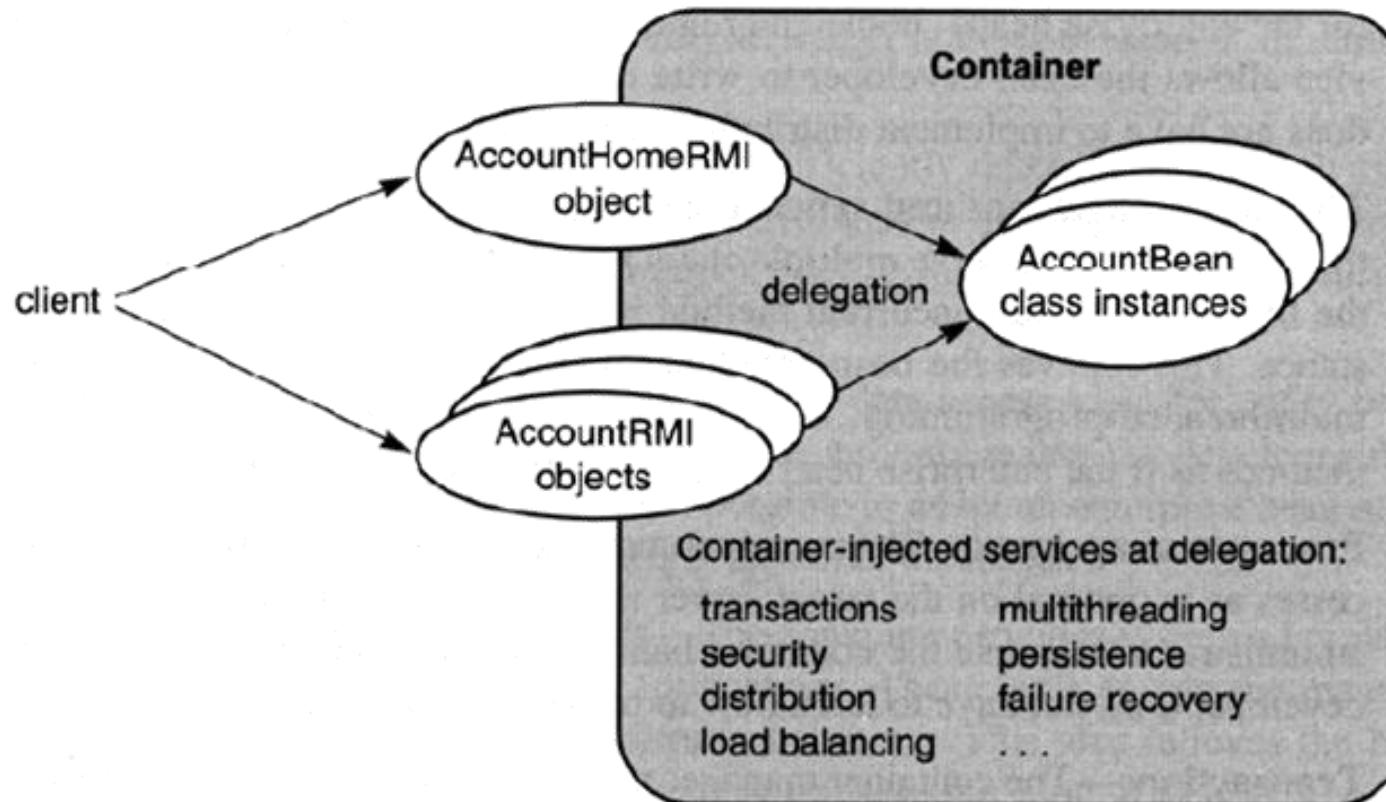
# Contenedor EJB

## Artefactos de contenedor



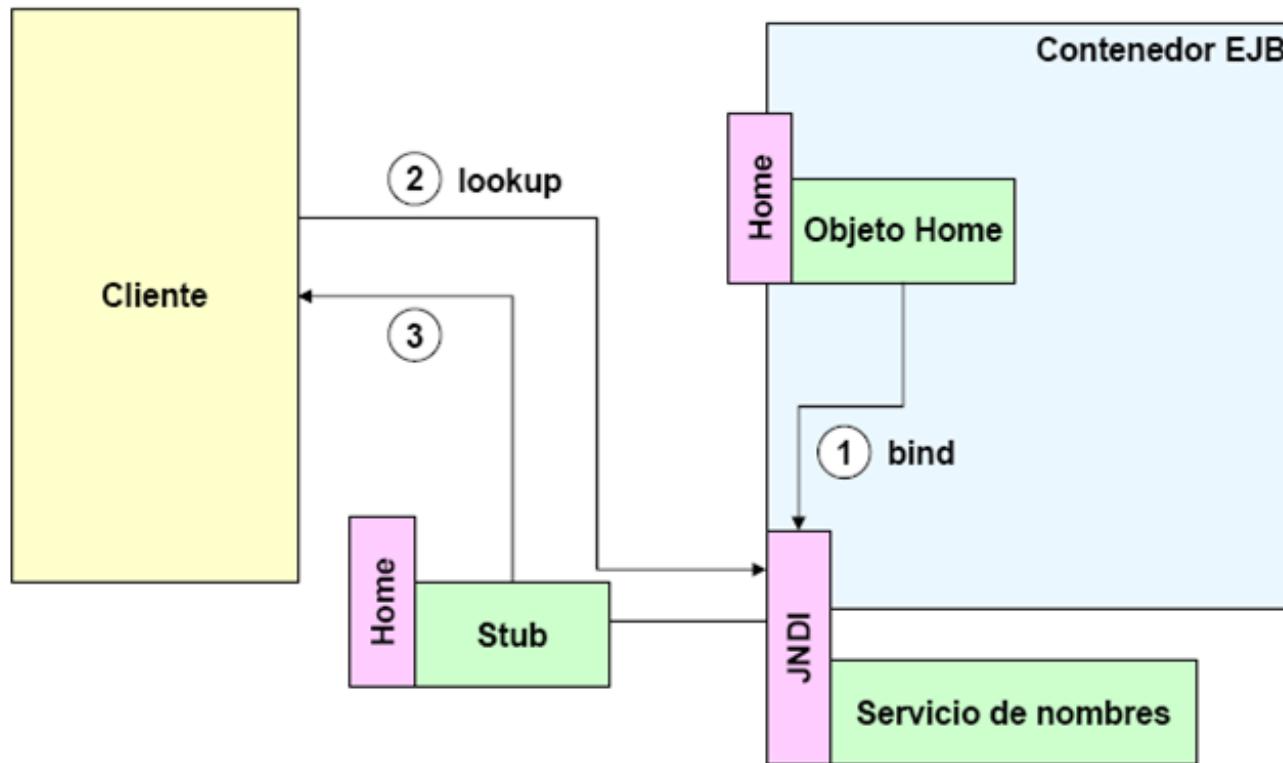
# Contenedor EJB

## Objetos RMI-IIOP



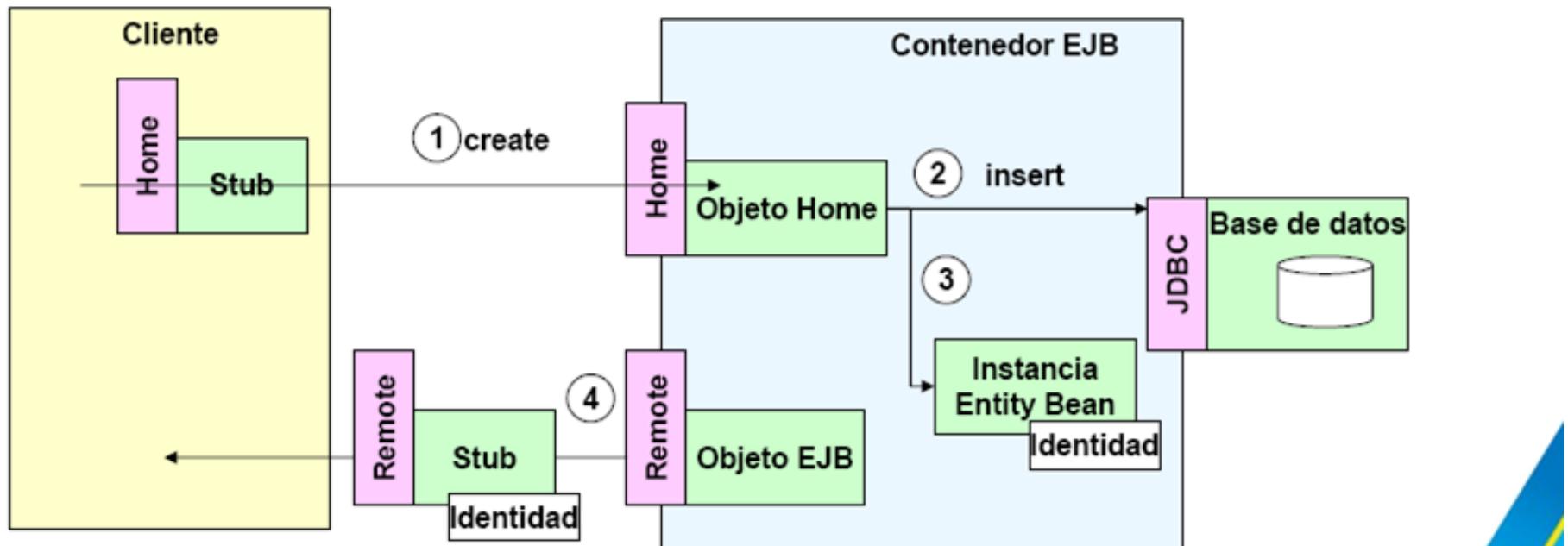
# Contenedor EJB

Llamada de un cliente: servicio de nombres



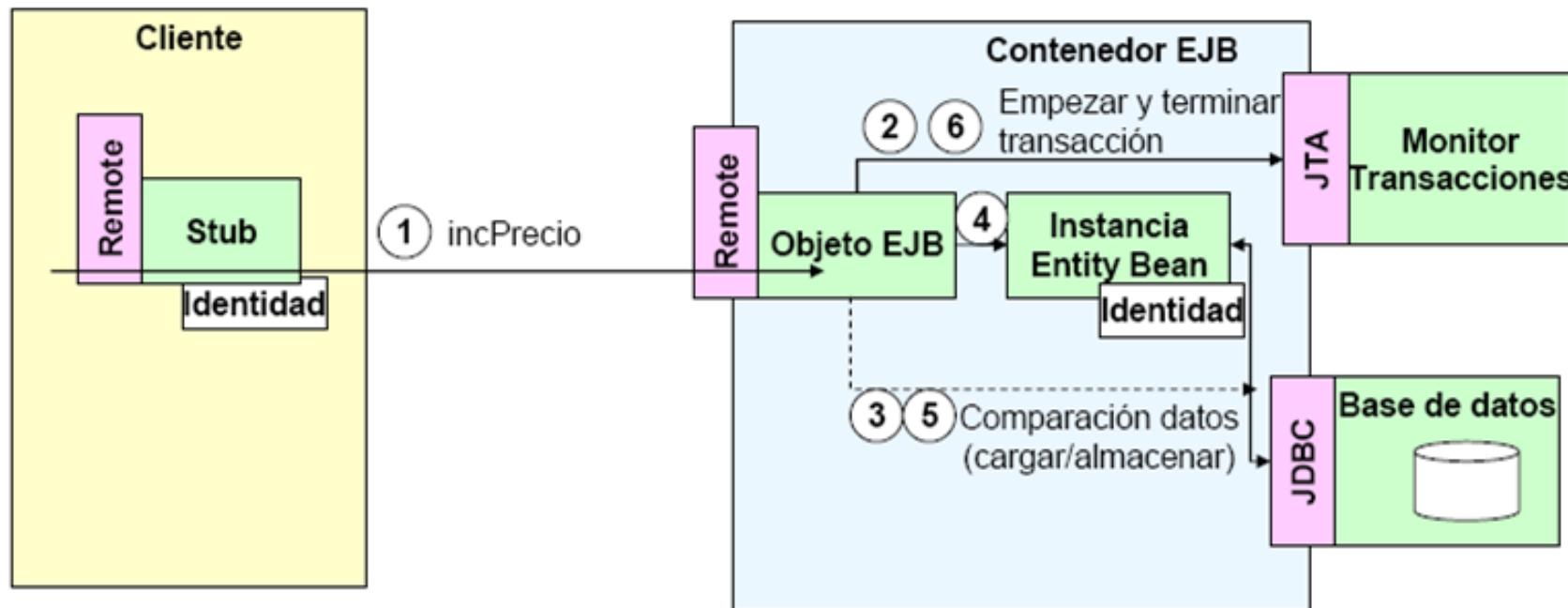
# Contenedor EJB

Llamada de un cliente: creación Entity Bean



# Contenedor EJB

## Llamada a un método de negocio



# Arquitectura EJB

## Empaquetamiento de EJBs

- Una aplicación J2EE se empaqueta en un archivo Enterprise Archive (EAR).
  - Un archivo EAR es un archivo Java Archive (JAR) con extensión ear.
- Se usa una herramienta de despliegue “Deployment Tool” como la del J2EE SDK para crear el archivo EAR
  - Descriptor de despliegue
  - Clases de los EJBs (interfaces e implementación)
  - Todos los JARs y/o WARs usados por la implementación del EJB

# Especificación EJB

## Creación de un EJB

- En este ejemplo el *enterprise bean* será un *stateless session bean* (*bean* de sesión “sin estado”) llamado ConverterEJB.
- Codificando el *enterprise bean*:
  - Remote interface
  - Home interface
  - Enterprise bean class

## Compilar

- Interface remota
  - (Converter.java),
- Interface home
  - (ConverterHome.java),
- Clase del enterprise bean
  - (ConverterBean.java)

# Especificación EJB

## Interfaz remota

```
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;
}
```

# Especificación EJB

## Interfaz Home

```
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;
}
```

# Especificación EJB

## SessionBean

```
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);
    }
}
```

# Especificación EJB

## SessionBean

```
public class ConverterBean implements SessionBean {  
    o o o  
    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);  
    }  
    o o o
```

# Especificación EJB

## SessionBean

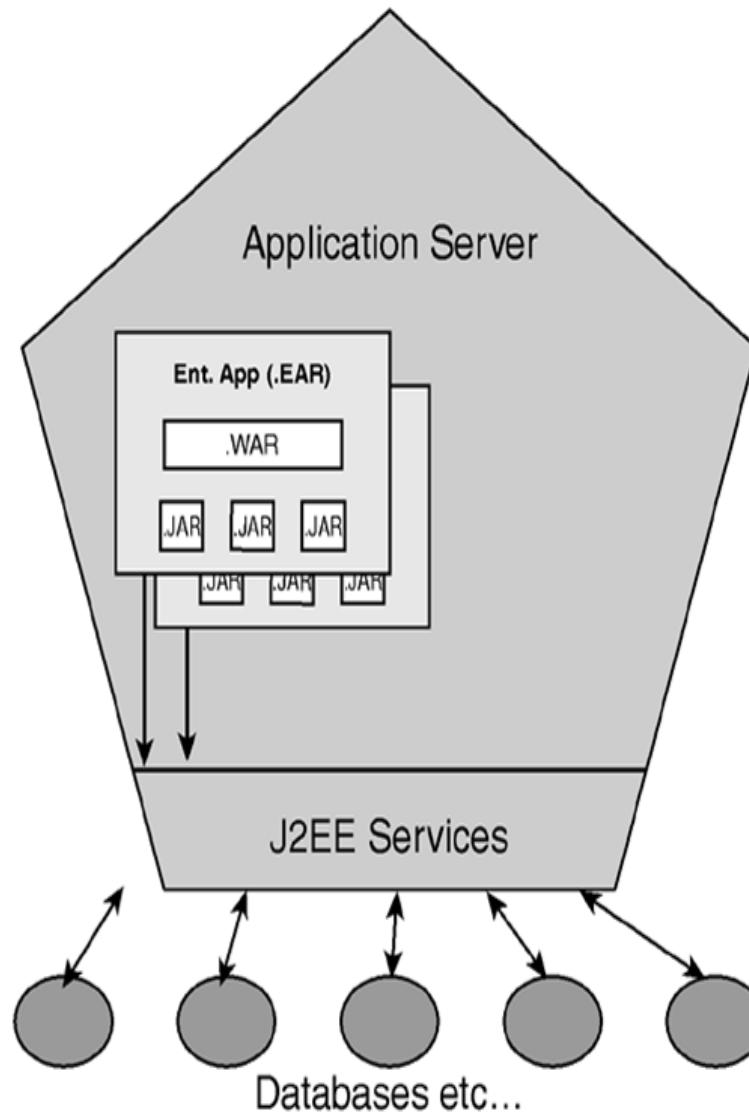
```
public class ConverterBean implements SessionBean {  
    o o o  
    public ConverterBean() {}  
    public void ejbCreate() {}  
    public void ejbRemove() {}  
    public void ejbActivate() {}  
    public void ejbPassivate() {}  
    public void setSessionContext(SessionContext sc) {}  
}
```

# Especificación EJB

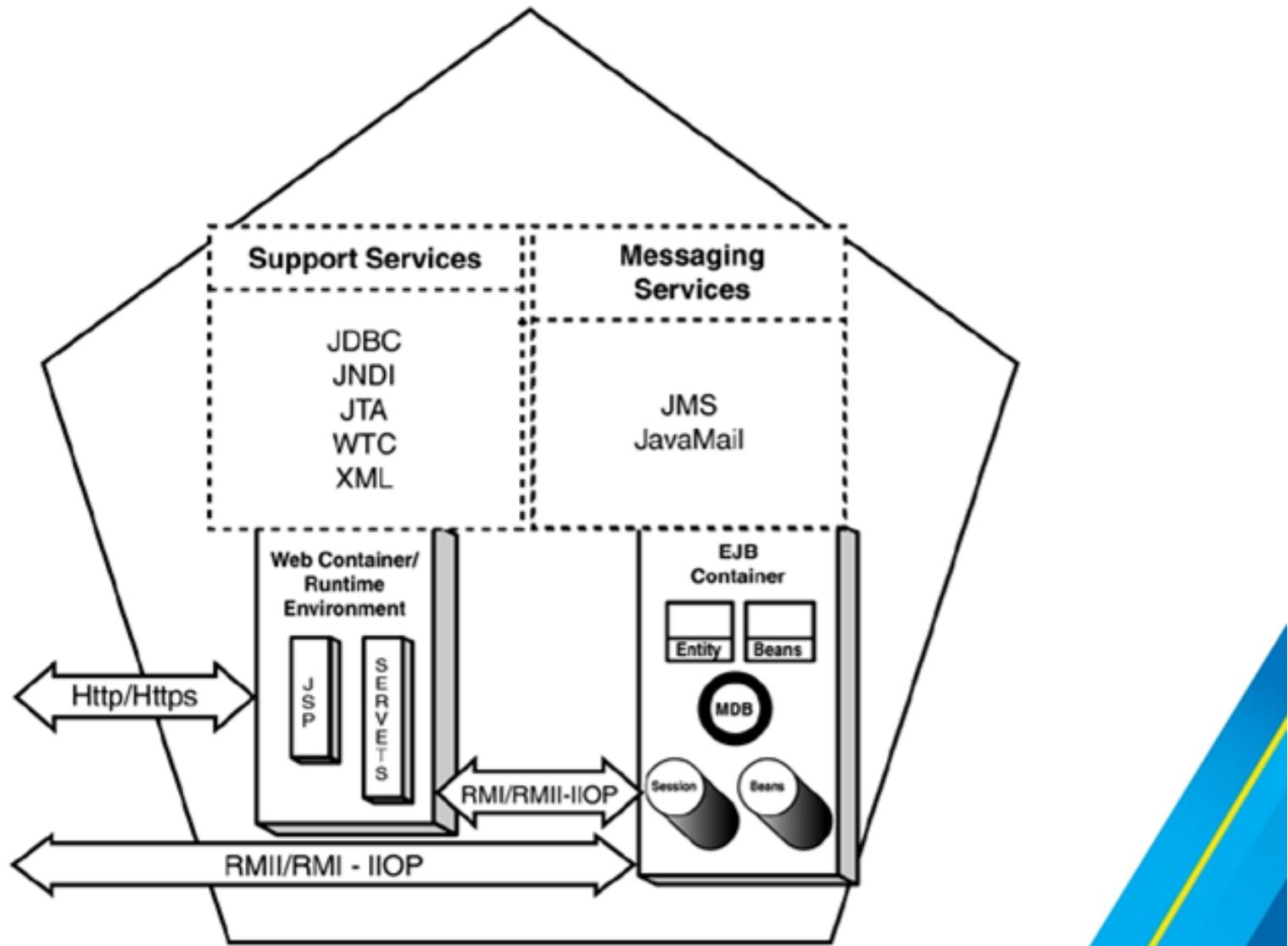
## Notas sobre el descriptor de despliegue

- Fichero independiente de la plataforma:
  - ejb-jar.xml
  - Contiene nombre de los EJBs, clases que implementan las interfaces y el bean, servicios “declarativos” requeridos...
- Fichero dependiente de la plataforma (p.ej weblogic-ejb-jar.xml):
  - Contiene información dependiente de la plataforma concreta. P. ej:
    - El nombre JNDI para cada interfaz home
    - Mapeo de los roles definidos en el ejb-jar a usuarios de la plataforma.
    - Nombres JNDI de otros recursos
    - Información de configuración de JavaMail

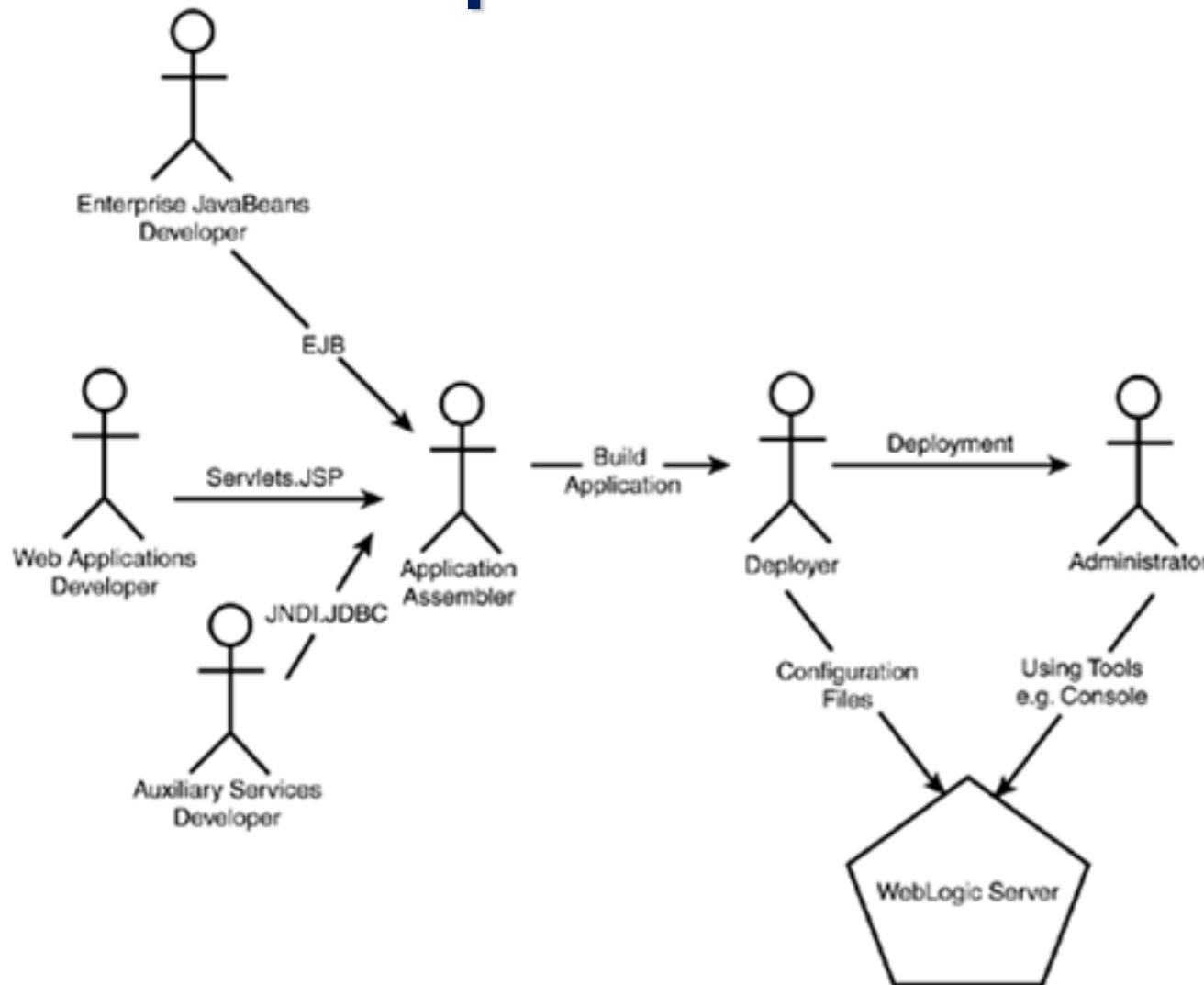
# Estructura básica de un servidor de aplicaciones



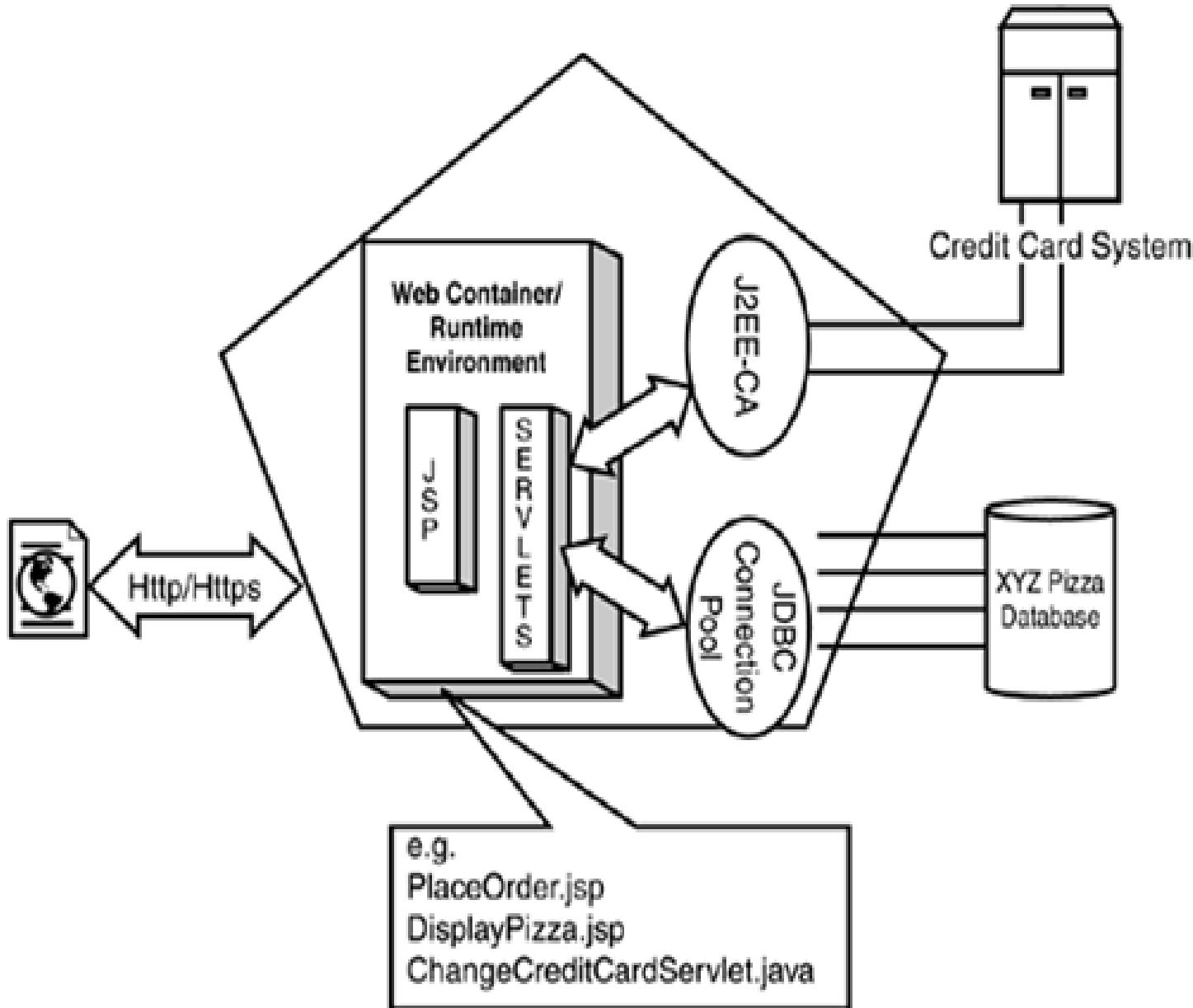
# Infraestructura de la plataforma Java EE



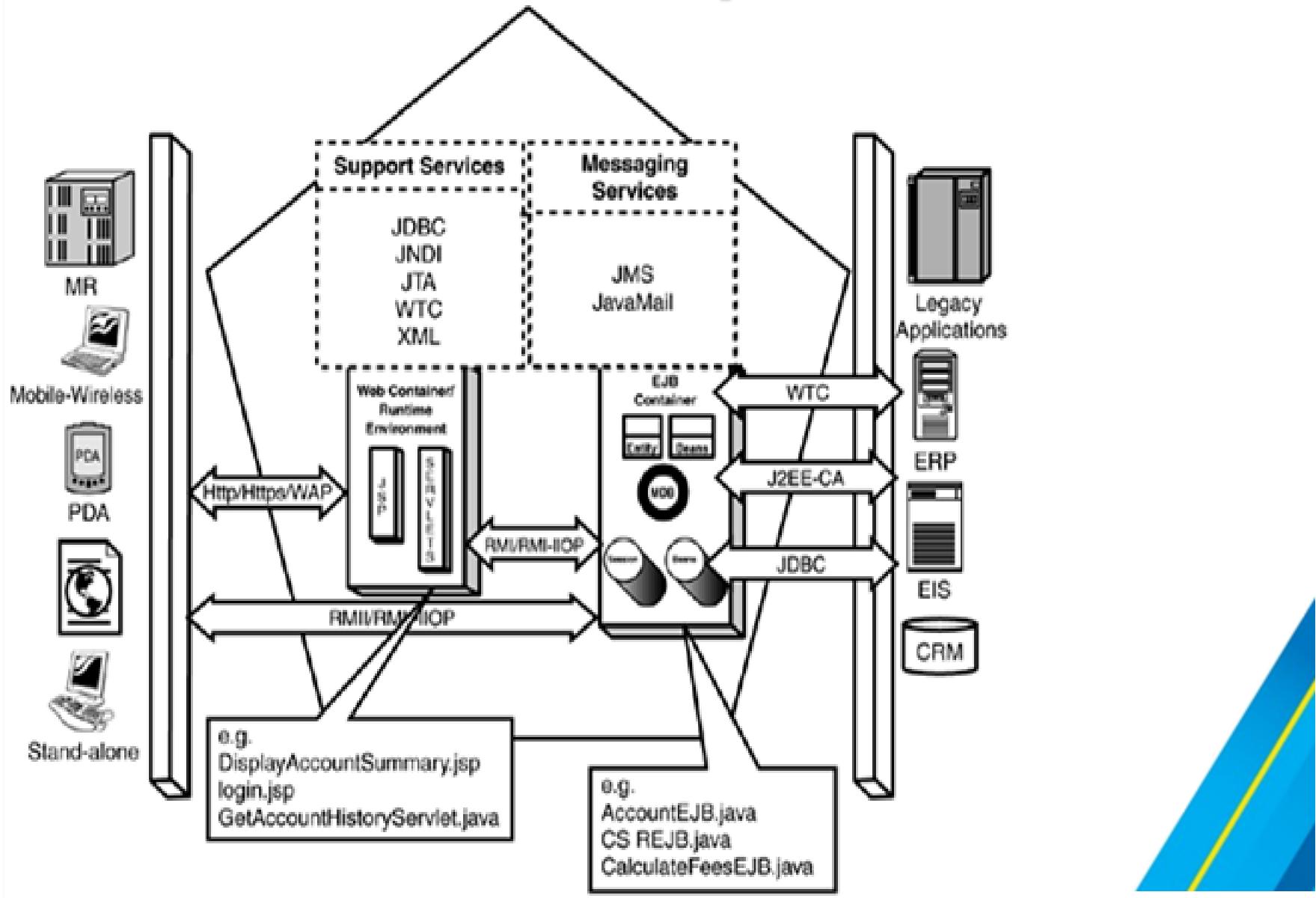
# Roles dentro del Desarrollo de aplicaciones Java EE

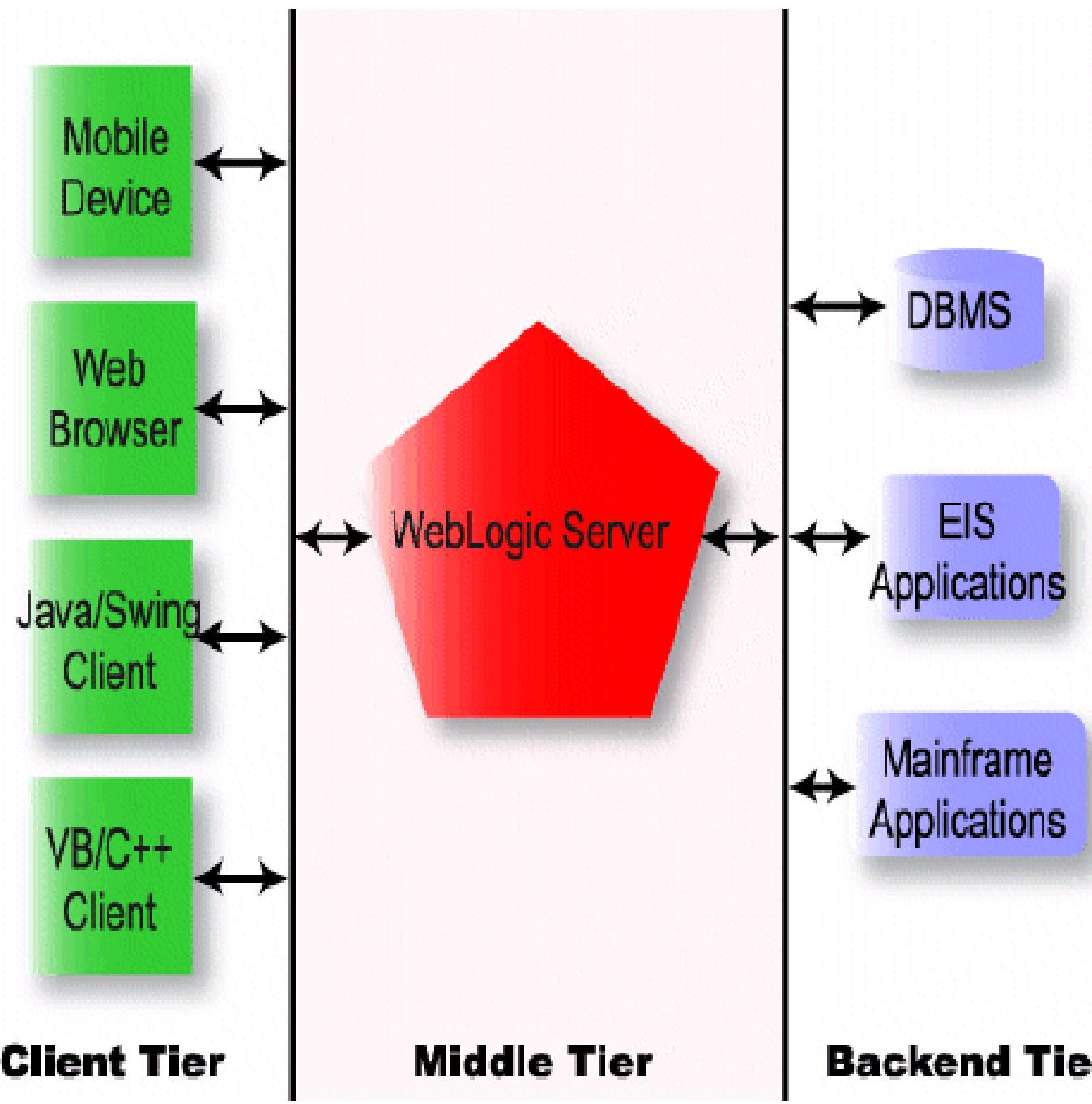


# ISIV Escenario de una Aplicación WEB



# Escenario de una Aplicación de n-capas





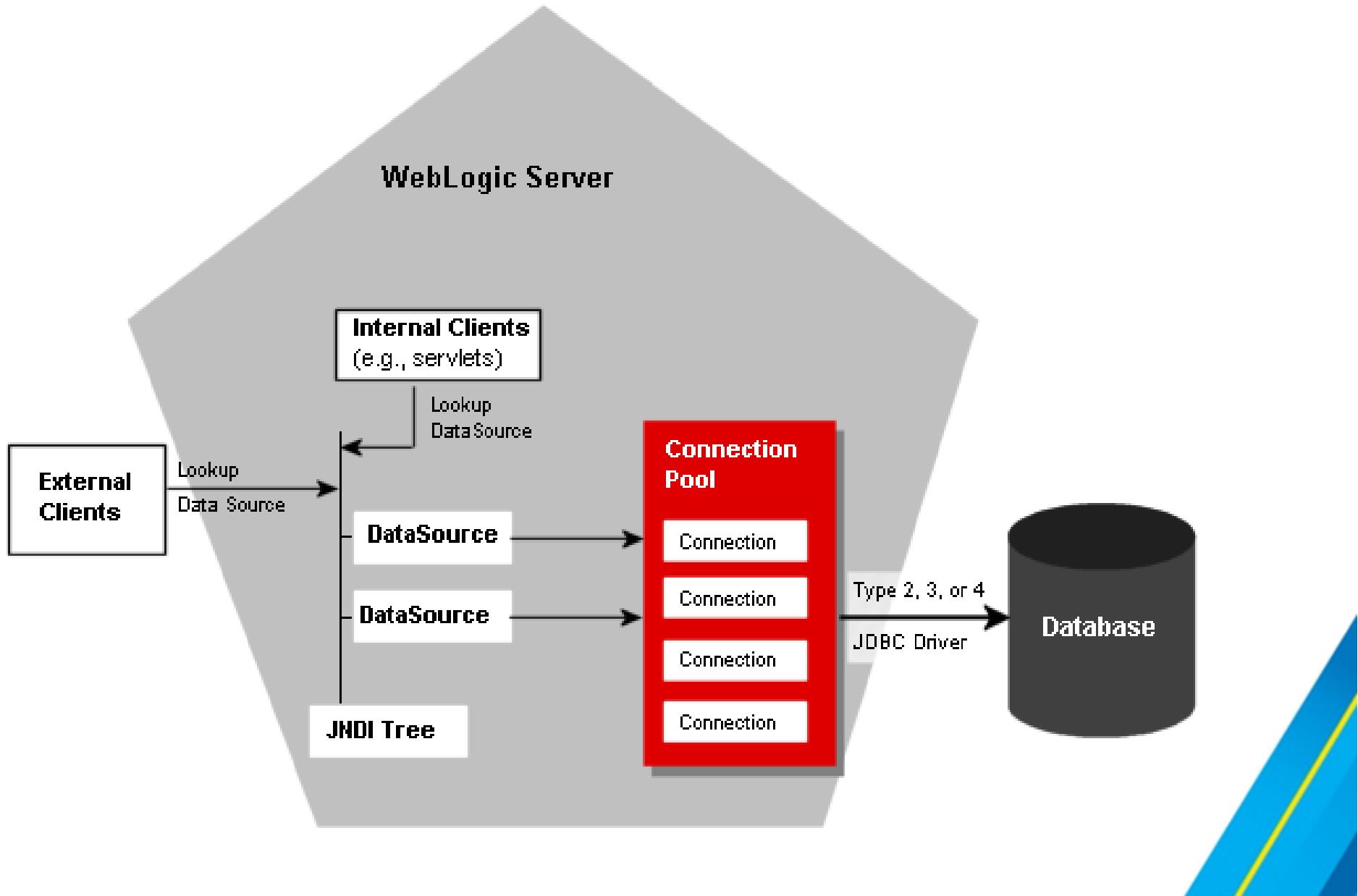
# Pool de Conexiones y los EJBs®

Edwin Maraví

[emaravi@cjavaperu.com](mailto:emaravi@cjavaperu.com)



# ISIV / Arquitectura del Pool de Conexiones



# JDBC Data Sources

- Un objeto DataSource permite a las aplicaciones JDBC obtener conexión al motor de base de datos de un pool de conexiones previamente establecido.
- Las aplicaciones buscan el Data Source en el árbol de JNDI y este apunta a un pool de conexiones



"GRACIAS"

