Transparencias

de

J2EE

Tema 4:
EJB

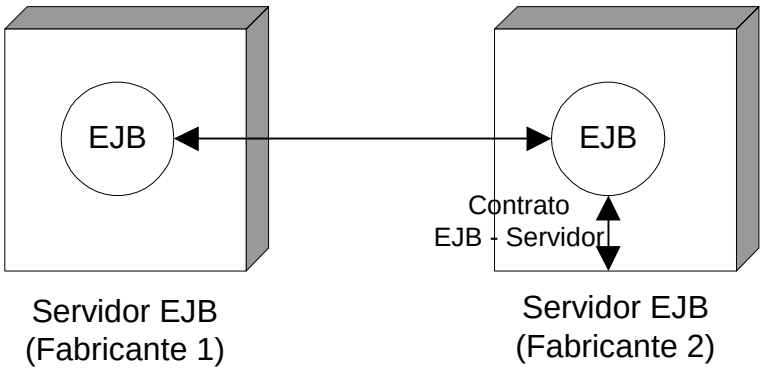
Uploaded by

Ingteleco

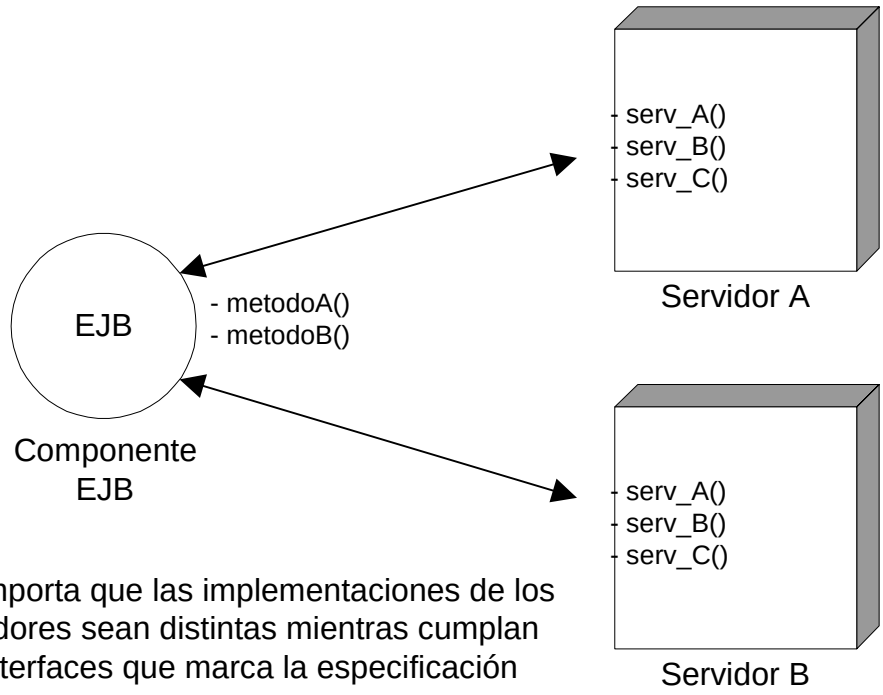
w TEMA 4: ENTERPRISE JAVABEANS

◆ INTRODUCCIÓN

- ✶ Enterprise JavaBeans (EJB)
  - ❑ Modelo de Componentes de Java
  - ❑ Componentes Java que se ejecutan en la parte servidora de una aplicación
  - ❑ Se ejecutan en el servidor de EJBs
  - ❑ Componentes destinados a soportar la LN de las aplicaciones
  - ❑ Fácil integración con CORBA ➔ Fácil integración con otros lenguajes
- ✶ Característica Destacada ➔ Framework de Servicios
  - ❑ Proporcionados por el servidor de EJBs
  - ❑ El desarrollador puede implementar sus aplicaciones de una forma muy sencilla haciendo uso de estos servicios.
  - ❑ El desarrollador sólo se centra en la lógica de negocio de los componentes. Los servicios ya están implementados, sólo hay que usarlos.
- ✶ Otras Características
  - ❑ Portabilidad
    - ■ Los EJBs se pueden ejecutar en cualquier servidor que cumpla la especificación



- Contrato Servidor – EJB
  - ✓ Los EJBs definen una serie de interfaces estándar para que las invoquen los servidores
  - ✓ Los servidores de EJB definen una serie de interfaces estándar para que sean invocados por los EJBs
  - ✓ Si tanto el desarrollador de EJBs como el fabricante de servidores EJB cumplen los contratos que marca la especificación ➔ Portabilidad Asegurada



No importa que las implementaciones de los servidores sean distintas mientras cumplan las interfaces que marca la especificación

Reusabilidad

- Los componentes EJB se implementan una vez y son reutilizados en múltiples aplicaciones que requieren su funcionalidad
- Desarrollo de Aplicaciones ➔ combinación de componentes reusables ya implementados
- Librerías de componentes
- Write Once, Run Anywhere

Roles en el desarrollo de aplicaciones

- Desarrollador de EJBs
  - ✓ Desarrollo de componentes
- Ensamblador de Componentes

- ✓ Combina componentes para formar aplicaciones

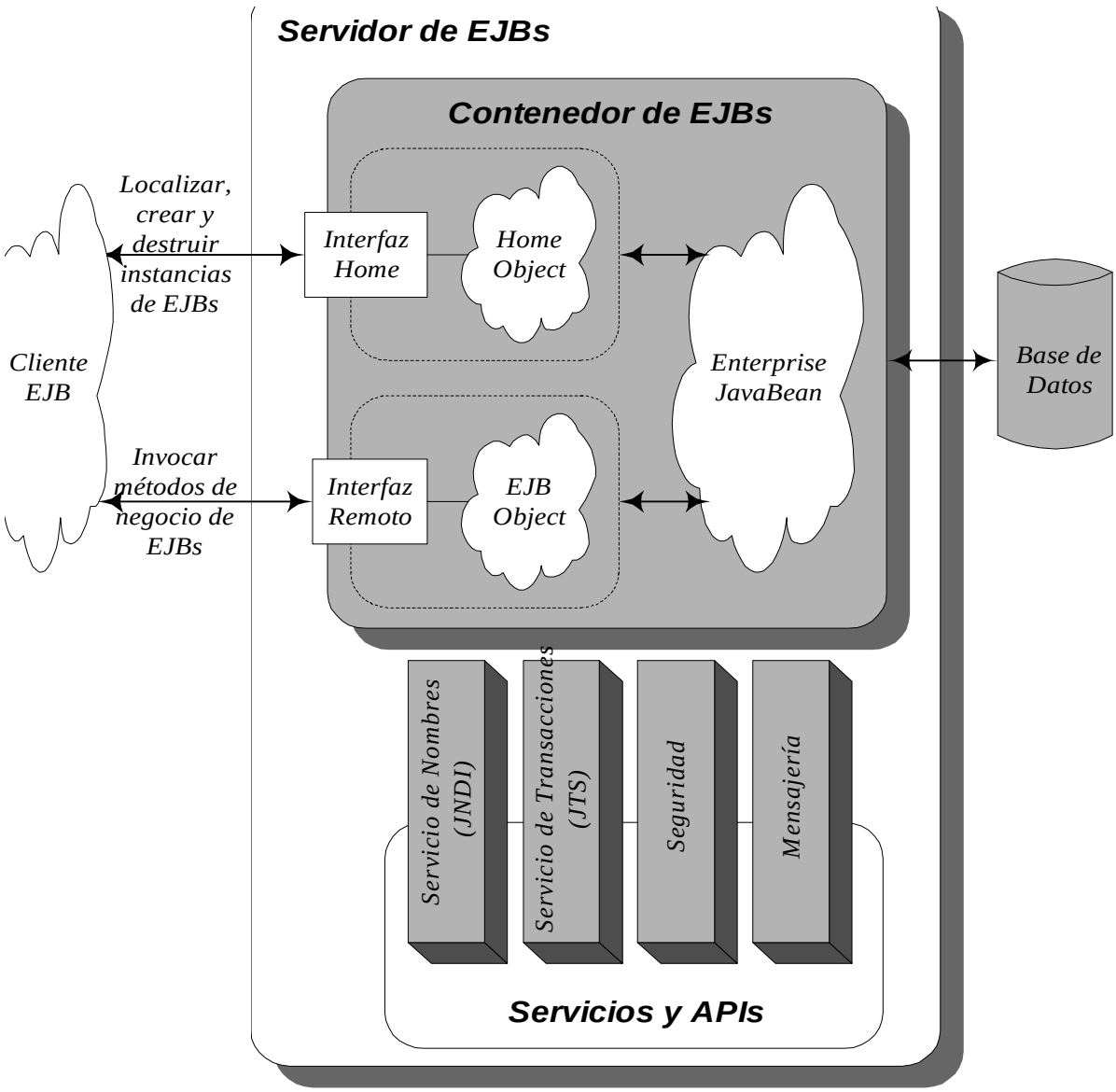
- Encargado del despliegue
  - ✓ Instala los componentes en el servidor de EJBs
- Compañía comercial
  - ✓ Desarrolla servidores de EJBs

- Soporte de Transacciones Distribuidas
  - Uno de los servicios del servidor de EJBs
  - Lo proporciona de forma transparente
- Integración con CORBA ➔ IIOP

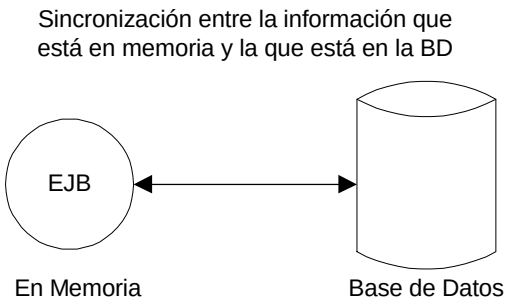
◆ EL MODELO EJB

✶ Elementos del Modelo:

- ❑ Servidor EJBs
- ❑ Contenedor EJBs
- ❑ HomeObject, EJBObject, Enterprise JavaBean
- ❑ Interfaz Home, Interfaz Remoto
- ❑ Cliente EJB
- ❑ Servicios



- ✶ Servidor EJBs
  - ❑ Entorno de ejecución para uno o más contenedores
  - ❑ Proporciona una serie de servicios accesibles a los contenedores
- ✶ Contenedor EJBs
  - ❑ Entorno de ejecución para los componentes EJB
  - ❑ Gestiona la ejecución de los EJBs y les proporciona una serie de servicios:
    - ■ Gestión del Ciclo de Vida
      - ✓ Creación y destrucción de instancias
      - ✓ Asignación y liberación de recursos
      - ✓ Activación y desactivación de objetos
      - ✓ Activation ➔ Recuperar del almacenamiento secundario e instanciar en mem.
        ➔ Cuando vuelva a ser requerido
      - ✓ Passivation ➔ Salvar a un almacenamiento secundario
        ➔ Si lleva mucho tiempo sin usarse
        ➔ Se liberan recursos
    - ■ Gestión del Estado
      - ✓ Mantiene o elimina el estado del componente después de cada invocación
      - ✓ Depende del tipo de componente (Stateless vs Statefull)
    - ■ Gestión de Transacciones ➔ Componentes transaccionales
    - ■ Gestión de Persistencia
      - ✓ Determina cuándo el estado del EJB debe ser salvado o recuperado de la BD



- ✓ 2 modos: Gestionada por el Contenedor y Gestionada por el propio Bean

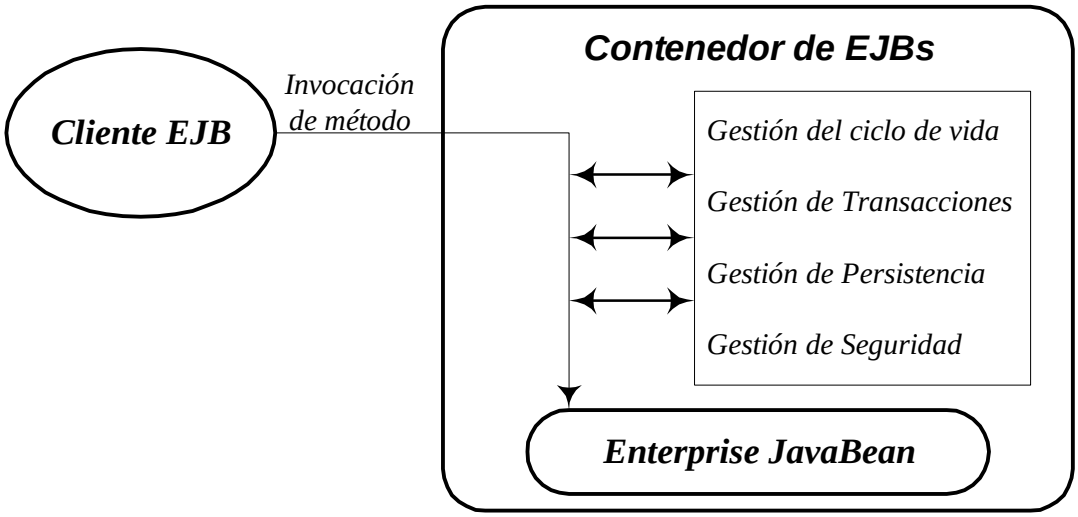
- CMP (Container Managed Persistence)
  - ✓ El contenedor decide cuándo recuperar o salvar un EJB en la BD
  - ✓ El contenedor se encarga de almacenar o recuperar la información de la BD (se encarga de los accesos a la BD)
  - ✓ Control total del contenedor
- BMP (Bean Managed Persistence)
  - ✓ El contenedor decide cuándo recuperar o salvar un EJB en la BD
  - ✓ El programador es quien realiza los accesos a la BD (el contenedor invoca el código del programador)
  - ✓ Control parcial del contenedor

Seguridad

- Chequeos en el acceso de los clientes a los EJBs

¿Cómo proporciona el contenedor estos servicios?

- No trata a todos los componentes por igual
- En el momento del despliegue se le indica al contenedor cómo debe tratar a cada componente ➔ *Deployment Descriptor*
- El contenedor no permite el acceso directo a los componentes desde el exterior
- El contenedor intercepta todas las invocaciones de métodos que van dirigidas hacia los EJBs



- Se asegura de que todas las llamadas pasan a través de él.
- Se encarga de proporcionar los servicios que requiere cada invocación de forma transparente

❑ Contratos

- Contrato entre Contenedor y EJB
  - ✓ Los EJBs poseen una serie de métodos *callback* que permiten al contenedor informar de eventos al EJB
  - ✓ Ejemplo: le informa de que va a ser recuperado o almacenado en la BD
- Contrato entre Contenedor y Servidor EJB
  - ✓ Leer en apuntes
- Contrato entre Contenedor y Cliente EJB
  - ✓ Leer en apuntes

✶ Interfaz Home y Objeto HomeObject

❑ Interfaz Home

- Tiene métodos que permiten crear, buscar y destruir instancias de un EJB
- Métodos relacionados con el ciclo de vida
- Todos los EJBs deben tener un interfaz Home
- Lo debe definir el desarrollador

❑ Objeto HomeObject

- Objeto que implementa el interfaz Home de un EJB
- Generado automáticamente por el contenedor durante el despliegue del componente
- Existe 1 por cada tipo de componente ➔ Compartido por todas las instancias del componente

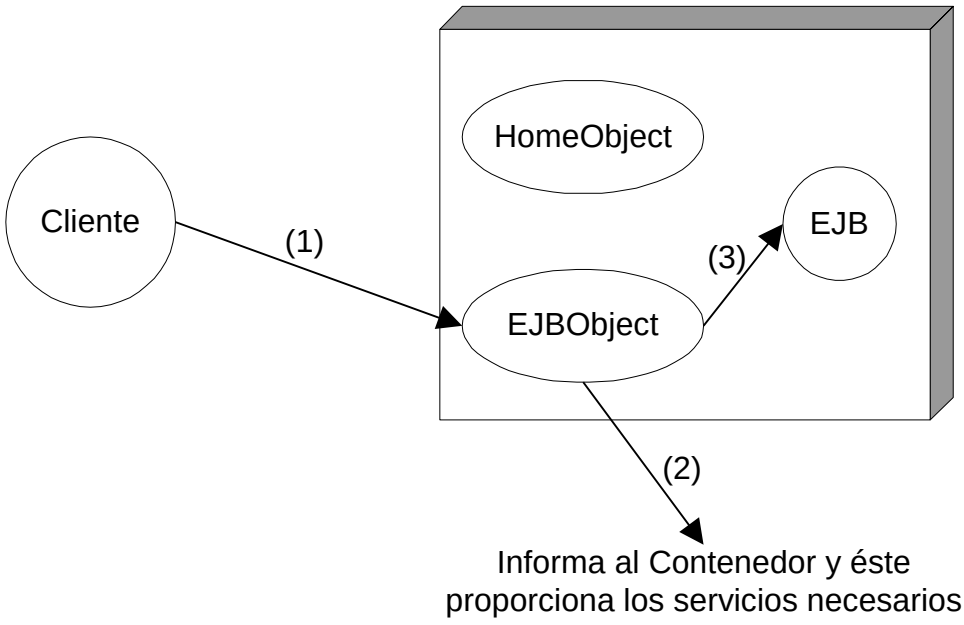
✶ Interfaz Remoto y EJBObject

❑ Interfaz Remoto

- Métodos de negocio del EJB

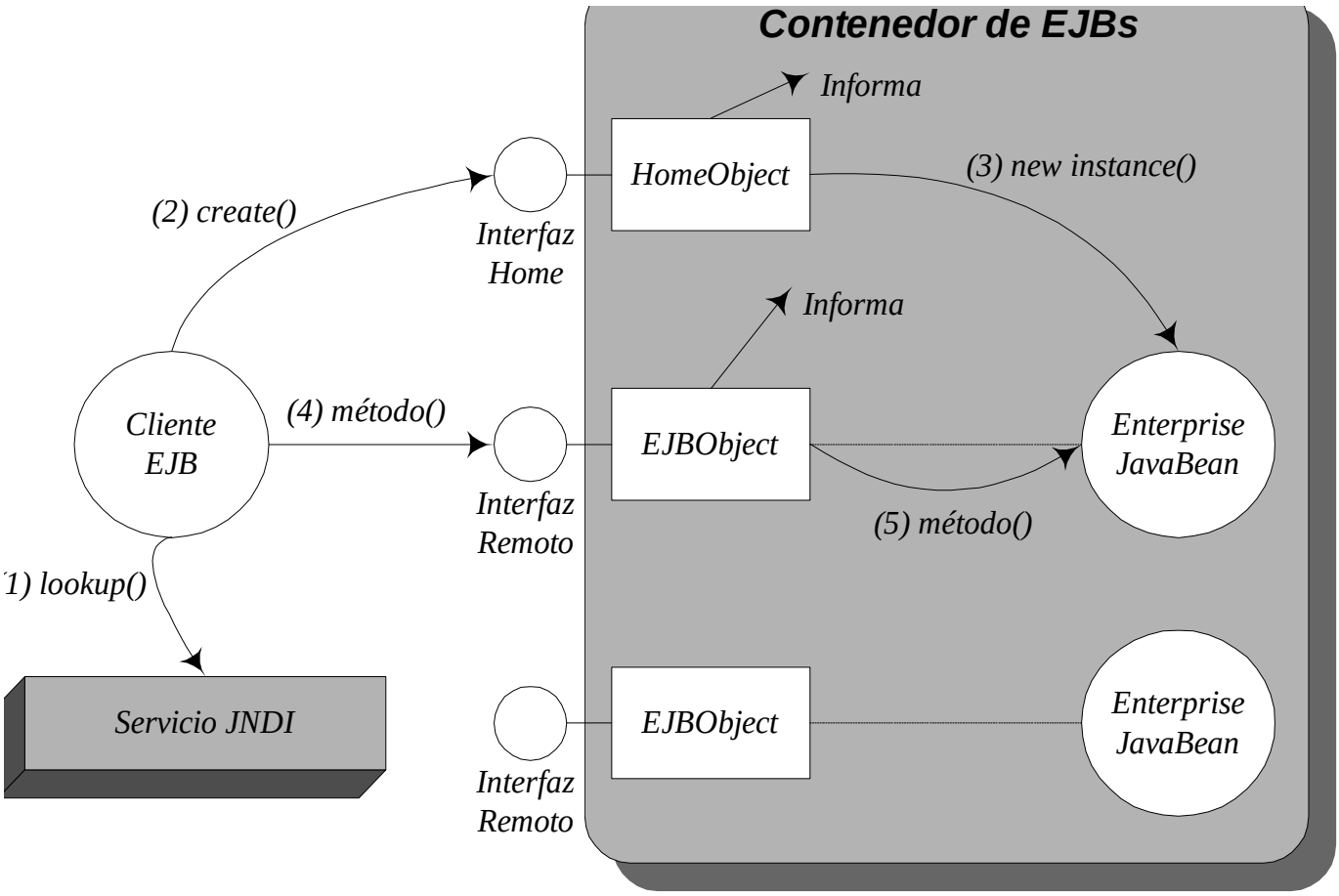
- Definido por el desarrollador
- El cliente sólo puede invocar los métodos que están en el interfaz remoto

- Cuando un cliente quiere usar un EJB...
  - Crear o Localizar una instancia del componente ➔ Mediante el interfaz Home
  - Obtener una referencia al interfaz Home ➔ usar sus métodos para crear o buscar instancias
- Objeto EJBObject
  - Objeto que implementa el interfaz remoto del componente
  - Generado automáticamente por el contenedor
  - Existe 1 por cada instancia del componente
- Cliente quiere invocar métodos de un EJB ➔ obtiene una referencia a su interfaz remoto ➔ invoca sus métodos
- El cliente nunca invoca directamente a un componente, siempre lo hace a través de su interfaz remoto
- ¿Cómo intercepta el contenedor los invocaciones?
  - Mediante los objetos EJBObject y HomeObject



- ✶ Enterprise JavaBean
  - ❑ Clase que define el componente EJB
  - ❑ Implementa la funcionalidad del EJB
    - ■ Métodos de negocio del interfaz remoto
    - ■ Métodos *callback* (contrato EJB-Contenedor)
- ✶ El Cliente EJB
  - ❑ Cualquier entidad que accede a los servicios de un EJB
  - ❑ Procedimiento:
    - ■ Recodatorio:
      - ✓ Interfaz Home ➔ Crear, buscar y destruir instancias
      - ✓ Interfaz Remoto ➔ Métodos de negocio
    - ■ JNDI ➔ Obtiene una referencia al interfaz Home (HomeObject)
    - ■ A través del interfaz Home ➔ Crea instancias de EJBs o buscar referencias al interfaz remoto de algún EJB
    - ■ A través del interfaz remoto ➔ se invocan los métodos del EJB

✶ Interacción entre un Cliente y un Componente EJB



◆ COMPONENTES EJB

- ✶ Elementos implementados por el Desarrollador:
  - ❑ Interfaz Home
  - ❑ Interfaz Remoto
  - ❑ Clase EJB
- ✶ Resto de Elementos ➔ Los proporciona el contenedor
- ✶ Ejemplo Adder ➔ EJB que suma números
  - ❑ Interfaz Home

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

public interface AdderHome extends EJBHome
{
   Adder create() throws RemoteException, CreateException;
   Adder create(int initial) throws RemoteException,
                                    CreateException;
}
```

  - ❑ Interfaz Remoto

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

public interface Adder extends EJBObject
{
   void add(int number) throws RemoteException;
   int getTotal() throws RemoteException;
}
```

- Clase EJB

```
import javax.ejb.*;

public class AdderEJB implements SessionBean
{

   int total;

   public void ejbCreate() {
      total = 0;
   }

   public void ejbCreate(int initial) {
      total = initial;
   }

   public void add(int number) {
      total += number;
   }

   public int getTotal() {
      return total;
   }

   public AdderEJB() {}

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

} // AdderEJB
```

✶ Tipos de Componentes EJB

- Dos tipos de componentes:
  - Session Bean
  - Entity Bean
- Entity Beans
  - Representan datos almacenados en una BD
  - Son Beans persistentes ➔ sus datos se almacenan en una BD
  - 1 instancia de un EntityBean = 1 fila de una tabla de BD
  - Su vida se extiende más allá que la aplicación ➔ sus datos quedan almacenados en la BD
  - Pueden ser compartidos por varios clientes
    - ✓ Todo cliente puede localizar un bean en la BD
    - ✓ Todo cliente puede usarlo
  - Deben poseer una clave primaria
    - ✓ Le distingue del resto de instancias de ese EntityBean
    - ✓ Debe ser un objeto (no tipo primitivo) ➔ String
  - Necesitan control de persistencia ➔ para mantener sincronizada su información con la de la BD
  - 2 modos de control de persistencia ➔ CMP vs BMP
  - CMP (Container Managed Persistence)
    - ✓ Persistencia gestionada totalmente por el contenedor
    - ✓ Decide cuándo salvar y recuperar el estado de la BD
    - ✓ Realiza los correspondientes accesos a la BD
    - ✓ El programador no tiene que escribir ni una línea de código de acceso a BD
  - BMP (Bean Managed Persistence)
    - ✓ Persistencia gestionada a medias entre el contenedor y el bean (programador)
    - ✓ Contenedor ➔ Decide cuándo salvar y recuperar el estado de la BD

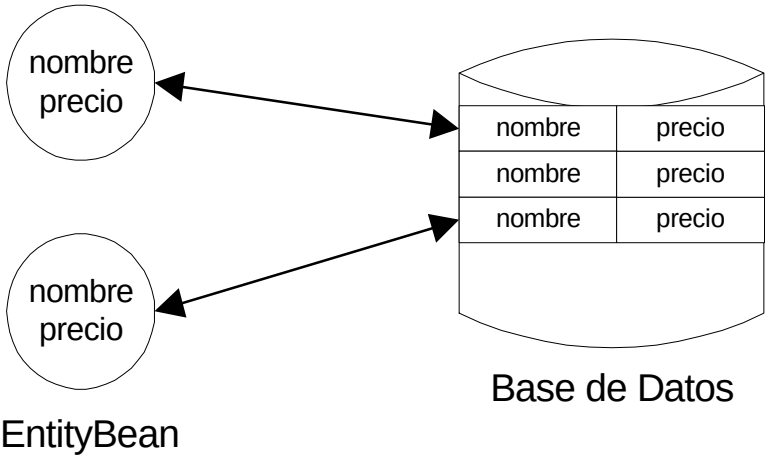
- ✓ Bean ➔ Realiza los accesos a la BD

  ➔ Contiene código JDBC para salvar y recuperar su estado de la BD

- ✓ El contenedor cuando decide que hay que recuperar o salvar la información de la BD, invoca al código JDBC del programador (bean)

- Ejemplos: cualquier entidad cuya información deba guardarse en una BD

  - ✓ Productos, Clientes, Reservas,...



Session Beans

- No representan datos almacenados en una BD ➔ No contienen información que se almacene en la BD
- No son persistentes
- Representan procesos de negocio ➔ Tareas que se ejecutan a petición de un cliente

  - ✓ La importancia de los Entity Bean radica en sus datos, mientras que la de los Session Bean radica en sus procesos (sus funciones)

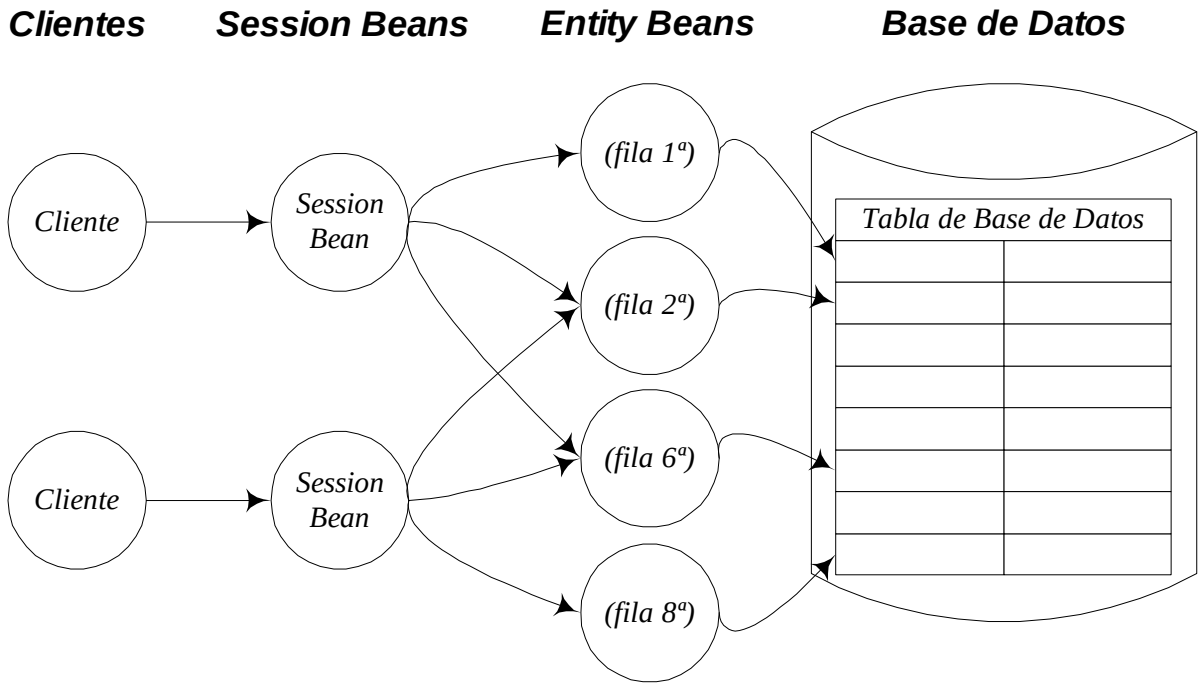
- Son una extensión de un cliente

  - ✓ No es compartido entre varios clientes
  - ✓ Contiene información de estado relativa al cliente
  - ✓ Son creados por un cliente y usados por ese cliente
  - ✓ Su vida se suele reducir a la vida del cliente

- Dos tipos de Session Bean: Stateless y Stateful

- Stateless Session Bean
  - Session Bean sin estado (no cambian nunca de estado)
  - No mantienen el estado entre invocaciones de métodos
    - ✓ No recuerdan nada de una invocación a la siguiente
    - ✓ Invocaciones independientes unas de otras
    - ✓ El contenedor borra la información de estado al final de cada invocación ➔ mismo estado al principio de cada invocación
  - Todas las instancias son idénticas ➔ todas proporcionan el mismo servicio ➔ pool de beans compartidos
  - Ejemplo: bean sin atributos (sólo métodos)
- Stateful Session Bean
  - Session Bean con estado
  - Mantienen el estado de una invocación a la siguiente
  - Su estado ➔ almacena información relativa al cliente
  - Ejemplos:
    - ✓ Cesta de la compra, entidad que hace reservas de vuelos,...

★ Diferencias Session Bean y Entity Bean



◆ IMPLEMENTACIÓN DE SESSION BEANS

✶ Stateless Session Bean

- Bean Converter ➔ Conversiones Monetarias
  - Dólares – Yens
  - Yens – Dolares
- Bean sin estado ➔ ni siquiera tiene atributos ➔ Stateless
- Sus métodos ➔ con los mismos parámetros ➔ siempre el mismo resultado
- Interfaz Home ➔ Nada nuevo

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

public interface ConverterHome extends EJBHome {

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

- Interfaz Remoto ➔ Nada nuevo

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

public interface Converter extends EJBObject {

   public double dollarToYen(double dollars)
                               throws RemoteException;
   public double yenToEuro(double yen)
                               throws RemoteException;
}
```

- Clase EJB

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

public class ConverterEJB implements SessionBean {

   // Implementación de los métodos del interfaz remoto
   public double dollarToYen(double dollars) {
      return dollars * 121.6000;
   }

   public double yenToEuro(double yen) {
      return yen * 0.0077;
   }

   // Constructor sin argumentos
   public ConverterEJB() {}

   // Implementación de los métodos callback
   public void ejbCreate() {}
   public void ejbRemove() {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public void setSessionContext(SessionContext sc) {}
}
```

- Todo Session Bean implementa el interfaz javax.ejb.SessionBean
- javax.ejb.SessionBean define una serie de *métodos callback* para los Session Beans que debemos implementar:
  - ✓ ejbActivate()➔ Es invocado por el contenedor justamente después de que el bean sea traído a memoria desde el almacenamiento secundario donde fue desactivado.
  - ✓ ejbPasivate()➔ Invocado justo antes de ser desactivado de memoria
  - ✓ ejbRemove()➔ Invocado justo antes de que el bean sea destruido
  - ✓ setSessionContext()➔ Invocado justo después de que el bean sea creado. El EJB recibe como parámetro un objeto SessionContext que permite al EJB comunicarse con el contenedor.
- Estos métodos son invocados por el contenedor para informar al bean de ciertos eventos que ocurran a lo largo de su ciclo de vida.

- El desarrollador introduce el código que quiere que se ejecute en cada caso.

✶ Stateful Session Bean

- Bean Adder ➔ Sumador visto anteriormente
- Bean con estado ➔ Atributo TOTAL
  - Atributo TOTAL
  - Constituye su estado interno
  - Stateful Session Bean
- Si comparamos el código del bean Adder (Stateful Session Bean) y el del bean Converter (Stateless Session Bean), veremos que no se diferencian en nada.
- ¿Diferencia Stateful Session Bean y Stateless Session Bean?
  - No implica ninguna diferencia a la hora de implementar (código)
  - Es una característica que se determina en el despliegue del componente
  - Despliegue ➔ se indica al contenedor como queremos que gestione el estado de nuestro componente

◆ IMPLEMENTACIÓN DE ENTITY BEANS

✶ Entity Bean con CMP

- ❑ Bean Product ➔ representa los productos de una BD
- ❑ Product (atributos)
  - ■ IdProducto
  - ■ Descripción
  - ■ Precio
- ❑ Interfaz Home

```
import java.util.Collection;
import java.rmi.RemoteException;
import javax.ejb.*;

public interface ProductHome extends EJBHome {

   public Product create(   String productId,
                            String description,
                            double balance)
                            throws    RemoteException,
                                      CreateException;

   public Product findByPrimaryKey(String productId)
                                  throws  FinderException,
                                          RemoteException;

   public Collection findByDescription(String description)
                                  throws  FinderException,
                                          RemoteException;

   public Collection findInRange(double low, double high)
                                  throws  FinderException,
                                          RemoteException;
}
```

Los entity beans son entidades que residen en la BD, son entidades compartidas entre varios cliente y por tanto, necesitamos métodos que permitan localizar instancias ya existentes

- Métodos de Búsqueda
  - ✓ Permiten buscar instancias ya existentes a partir de una serie de criterios
  - ✓ Signatura:

```
<Referencia_Interfaz_Remoto> findXXX (<atributo_bean>)
<Colección_referencias> findXXX (<atributo_bean>)
```

- Ejemplos:

`Product findByPrimaryKey(String PK)`

  - ✓ Localiza un EJB dada su clave primaria
  - ✓ Recibe como parámetro la clave primaria
  - ✓ Devuelve la referencia al interfaz remoto del EJB con esa PK ➔ sólo puede haber un EJB con esa PK
  - ✓ Todo entity bean debe implementar este método (obligatorio)

`Collection findByDescription(String desc)`

  - ✓ Localiza los EJBs que tengan una determinada descripción
  - ✓ Recibe como parámetro la descripción
  - ✓ Devuelve una colección de referencias al interfaz remoto de los EJBs que cumplan con esa descripción

- Todos los métodos de búsqueda lanzan la excepciones:
  - ✓ **java.rmi.RemoteException**
  - ✓ **javax.ejb.FinderException** ➔ Se produce si no existe ningún bean que satisface el criterio de búsqueda
- En CMP los accesos a la BD los realiza el contenedor
  - ✓ CREATE ➔ “insert...”
  - ✓ FINDXXX ➔ “Select…”
  - ✓ ¡La implementación de los métodos de búsqueda los realiza el contenedor de forma transparente, el programador sólo tiene que declararlos en el interfaz Home!

- Interfaz Remoto

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

public interface Product extends EJBObject {

   public void setPrice(double price)
                                    throws RemoteException;

   public double getPrice() throws RemoteException;

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

  - Métodos que permiten consultar y modificar la información del entity bean (get y set)
  - La importancia de los entity bean está en sus datos, por tanto, sus métodos de negocio están orientados a la manipulación de sus datos.

- Clase EJB

```
import java.util.*;
import javax.ejb.*;

public class ProductEJB implements EntityBean {

   public String productId;   //Primary Key
   public String description;
   public double price;

   private EntityContext context;

   //Implementación de los métodos de negocio del interfaz remoto
   public void setPrice(double price) {
      this.price = price;
   }

   public double getPrice() {
      return price;
   }

   public String getDescription() {
      return description;
   }
```

```
    //Implementación de los métodos del interfaz home
    public String ejbCreate( String productId, String description,
                             double price) throws CreateException {
       if (productId == null) {
          throw new CreateException("The productId is required.");
       }

       this.productId = productId;
       this.description = description;
       this.price = price;

       return null;
    }

    //Implementación de los métodos callback
    public void setEntityContext(EntityContext context) {
       this.context = context;
    }

    public void ejbActivate() {
       productId = (String)context.getPrimaryKey();
    }

    public void ejbPassivate() {
       productId = null;
       description = null;
    }

    public void ejbRemove() { }
    public void ejbLoad() { }
    public void ejbStore() { }
    public void unsetEntityContext() { }
    public void ejbPostCreate( String productId, String description,
                               double balance) { }

}
```

- Atributos que almacenan la información del producto ➔ coincidirán con los de la BD
- Clave Primaria
  - ✓ Debe ser un objeto (no un tipo primitivo)
  - ✓ Nosotros siempre usaremos un String
  - ✓ ProductId
- Implementa los métodos de negocio del interfaz remoto
- Proporciona un método **`ejbCreate()`** por cada método **`create()`** del interfaz Home

- Método **`ejbCreate()`**
  - ✓ No devuelve `void` como ocurría con los session bean
  - ✓ Devuelve el tipo de la clave primaria (`String`)
  - ✓ En el código, en CMP devolverá siempre null (el contenedor no lo usa para nada) y en BMP devolverá la clave primaria del componente
- Todo entity bean implementa el interfaz **`javax.ejb.EntityBean`**
- Este interfaz define una serie de *métodos callback* que deben ser implementados por la clase EJB
- Métodos Callback:
  - ✓ ejbActivate()
  - ✓ ejbPasivate()
  - ✓ ejbRemove()
  - ✓ ejbLoad(): invocado por el contenedor justo después de que el bean haya sido recuperado de la BD. Él recupera los datos de la BD y nos avisa.
  - ✓ ejbRestore(): invocado por el contenedor justo antes de que éste vaya a salvar los datos en la BD
  - ✓ setEntityContext(): similar al setSessionContext()
  - ✓ unSetEntityContext(): justo antes de que vaya a ser eliminado el bean
  - ✓ ejbPostCreate(): método opcional que debe emparejarse con algún método ejbCreate(mismos argumentos). Invocado después de la finalización del método create() relacionado
- No es necesario introducir código JDBC que realice los accesos a la BD ➔ Lo hace el contenedor
- No hay que implementar los métodos de búsqueda definidos en el interfaz Home ➔ los implementa el contenedor

- Entity Bean con BMP
  - Bean Account ➔ Representa cuentas bancarias almacenadas en una BD
  - Atención a:
    - Métodos ejbCreate(): poseen la “INSERT” de acceso a la BD
    - Implementación de los métodos de búsqueda mediante “SELECT”
    - Todos los accesos a BD mediante JDBC
  - No se usarán en el proyecto

◆ CLIENTES EJB

- Entidad software que accede a los servicios de un EJB
- Pasos:
  - Obtener una referencia al interfaz Home a través de JNDI
  - Crear o localizar instancias del componente
  - Invocar los métodos de esas instancias
- Obtener una referencia al interfaz Home a través de JNDI
  - El contenedor, durante el despliegue, registra el interfaz Home del componente en el servicio JNDI con un nombre
  - El cliente, si conoce el nombre, puede obtener una referencia al interfaz Home a través de JNDI
    - Crea un contexto JNDI
    - Usa el método lookup(“nombre”) y obtiene una referencia al interfaz Home
    - Convierte la referencia al tipo adecuado del interfaz ➔ ProductHome

```
Context initial = new InitialContext();
Object objref = initial.lookup("MyProduct");
ProductHome home=(ProductHome)PortableRemoteObject.narrow(objref,
                                                  ProductHome.class);
```

- Crear o Localizar instancias del componente
  - Usamos los métodos del interfaz Home para crear o localizar instancias (create o find)
  - Obtenemos una referencia al interfaz remoto del objeto creado o buscado ➔ Product

```
Product producto = home.create("456", "Wooden Duck", 13.00);
Product producto2 = home.findByPrimaryKey("876");
```

- Invocar los métodos de esas instancias
  - Invocamos los métodos de negocio del componente a través del interfaz remoto.

```
producto.setPrice(14.00);
double price = producto2.getPrice();
```