Programación con EJBs: Entity Beans y Session 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 el contrato existente entre el servidor de aplicaciones y los diferentes tipos de componentes
  - Comprender el esquema de concurrencia de cada componente
  - Comprender los esquemas de gestión de *pooling*, *swapping*, *activation* y *pasivación*
  - Comprender los modelos de gestión de la persistencia de tipo CMP y BMP
- Comprender y distinguir los ciclos de vida de los diferentes tipos de componentes EJB
  - *Entity Bean*, *Session Bean* (stateless, statefull) y *Message Beans*
  - Ver cómo los ciclos de la vida se corresponden al API
- Aprender a codificar EJB
  - Diseño de interfaces, métodos de negocio, modelo de persistencia e invocación desde clientes
    - Con Entity Beans de tipo CMP
    - Con Session Beans de tipo stateless

Índice (1/3)

Bloque I: Modelo ofertado por el contenedor a los EJBs

- Qué modelo de concurrencia se asigna a cada tipo de componente
  - El problema de la re-entrancia
- Cómo se gestiona las estancias de cada uno de los beans
  - Cómo se asigna una instancia del pool a un Entity Bean
  - El swapping de un stateless bean
  - La activación y pasivación de un session bean
- Cómo se gestiona la persistencia (Entity Beans)
  - Dos modelos de persistencia soportados (CMP y BMP)
  - Configuraciones más típicas (objeto-relacional, persistencia BD O-O y aplicación legado)
  - Detalles del desarrollo con CMP
  - Detalles del desarrollo con BMP

Índice (2/3)

Bloque II: Ciclos de vida y reflejo en el API

- Entity Beans
- Session Beans
- Message Beans

Bloque III: Programación con Entity Beans de tipo CMP

- Características del modelo CMP
- Características del modelo BMP
- Ejemplo del CabinBean (tipo CMP)
  - Interfaz CabinRemote
  - Interfaz CabinRemoteHome
  - Interfaz CabinBean
  - Cliente Remoto
- Detalles
  - Contexto inicial en JNDI
  - Métodos lookup
  - Métodos narrow
  - Consideraciones a tener en cuenta sobre RMI

Índice (3/3)

Bloque IV: Programación con Session Beans de tipo StateLess

- Características generales
- Diferencias existentes entre un StateFull y un StateLess
- Ejemplo del TravelAgentBean (tipo stateless)
  - Interfaz CabinRemote
  - Interfaz TravelAgentRemoteHome
  - Interfaz TravelAgentBean
  - Cliente TravelAgentClient

Bloque V: Ejercicios y cuestiones para profundizar

Modelos de concurrencia soportados

- En el caso de los Session Beans
  - El acceso concurrente no está soportado
    - bean de sesión con estado sirve únicamente un cliente
    - el ámbito de un bean de sesión sin estado es una sola invocación
- En el caso de los Entity Beans
  - El acceso concurrente es prohibido por defecto
- En el caso de los Message Beans
  - Permitido el tratamiento concurrente de mensajes (y esencial)
    - múltiples instancias
- Nota sobre los Session y Entity Beans
  - Creación de hilos por bean está prohibida
  - Permite que el contenedor mantenga control de concurrencia

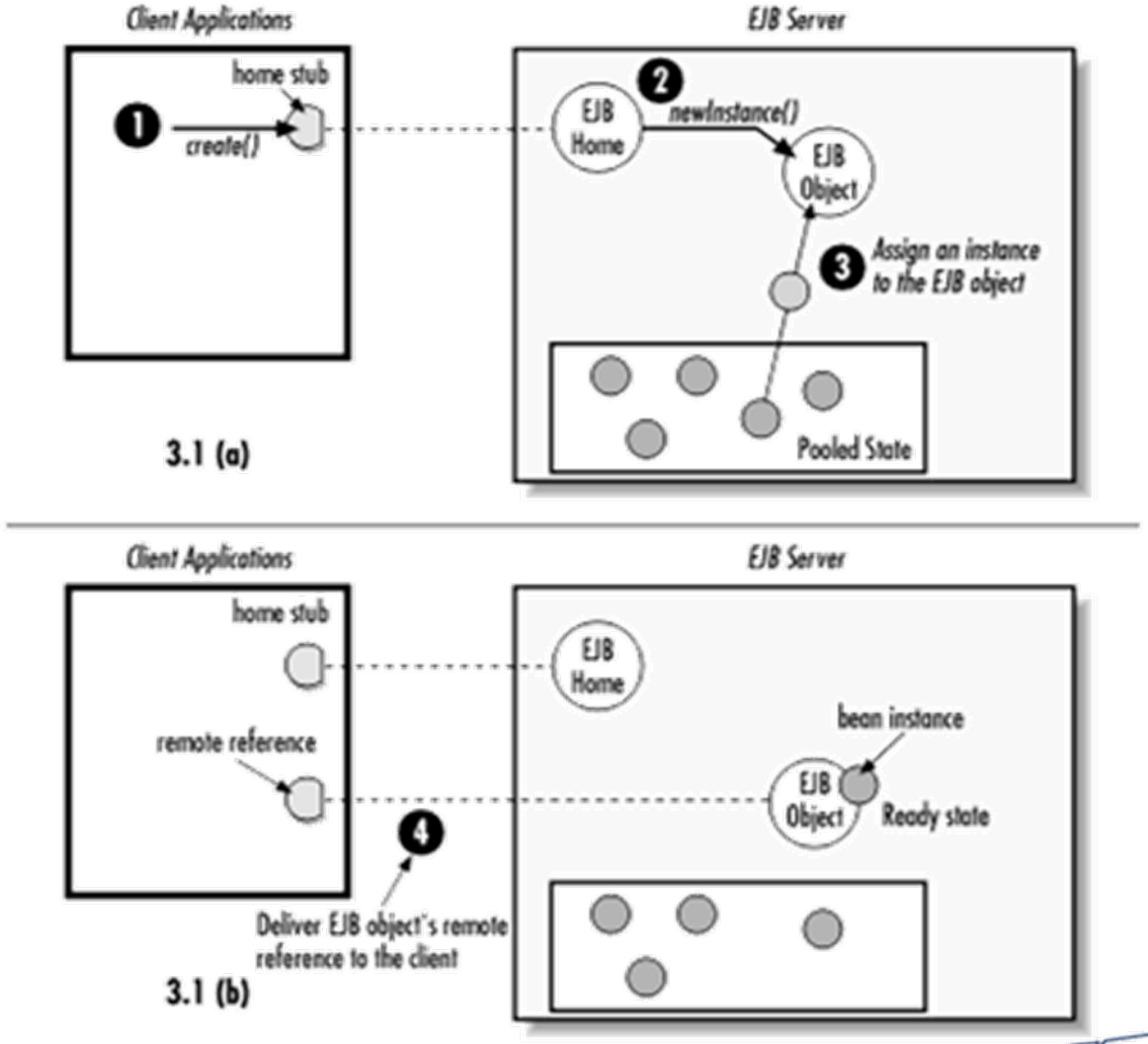
Concurrencia y código reentrante en EJBs

- Los beans de entidad no son reetrantes (por defecto)
  - No permite “loopbacks”: “callbacks” directos o indirectos
    - p.ej. un cliente llama al objeto A, A llama a B, B llama a A
  - en el descriptor de despliegue se puede cambiar:
    - `<reentrant>False</reentrant>`
- Los beans de sesión nunca pueden ser reentrantes
  - Una excepción es lanzada si un “loopback” es intentado
- Problema: contenedor no puede distinguir entre
  - Acceso concurrente por el mismo hilo via “*loopback*”
  - Acceso concurrente por distintos hilos
- Instancia de bean invoca sus propios métodos
  - No se considera reentrante

“Pooling” de instancias

- Delegación: los clientes no acceden a las instancias directamente
  - Acceso a través de objetos EJB
    - implementación de las interfaces locales y remotas (EJB 2.0)
- El servidor mantiene un pool de instancias
  - Que se asocian a los objetos EJB cuando se requiere
  - Aplica a beans de entidad y a beans de sesión sin estado
    - Puede ser compartido por múltiples clientes
  - Manejo de recursos más eficiente
- Beans dirigidos a mensaje
  - beans se suscriben a un destino de mensajes especifico
  - clientes entregan mensajes a alguno de los destinos
  - contenedor crea un “pool” de beans por cada destino

Asignación de una instancia del "Pool"



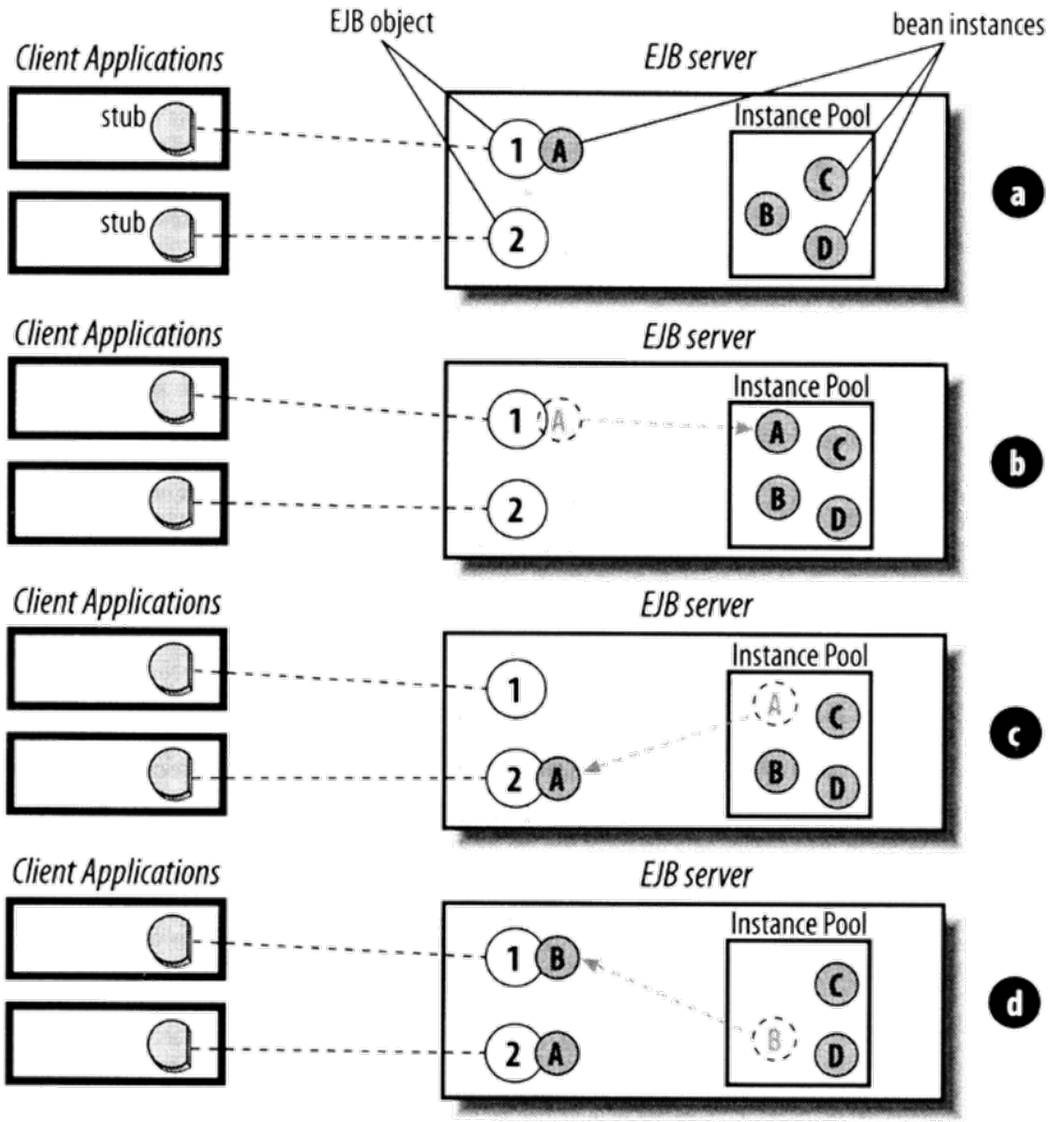
Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author)
Publisher: O'Reilly

Beans de sesión sin estado & "Swapping" de instancia

- Beans de sesión sin estado ("*stateless" session beans*)
  - Se declaran como tales en el descriptor de despliegue
    - se codifican de la misma forma que los beans de sesión con estado (*"stateful" session beans*)
  - No deben mantener el estado conversacional, aunque
    - pueden tener variables de instancia
    - pueden obtener información de JNDI ENC

- Asignación de una instancia por invocación
  - el mismo cliente servido por instancias de bean diferente
  - la misma instancia de bean es "*swapped*" entre objetos EJB / clientes
  - pocas instancias de beans de sesión sin estado sirven a muchos clientes
  - ciclo de vida: combina los estados "*ready*" y "*pooled*" de los beans de entidad

Ejemplo Swapping

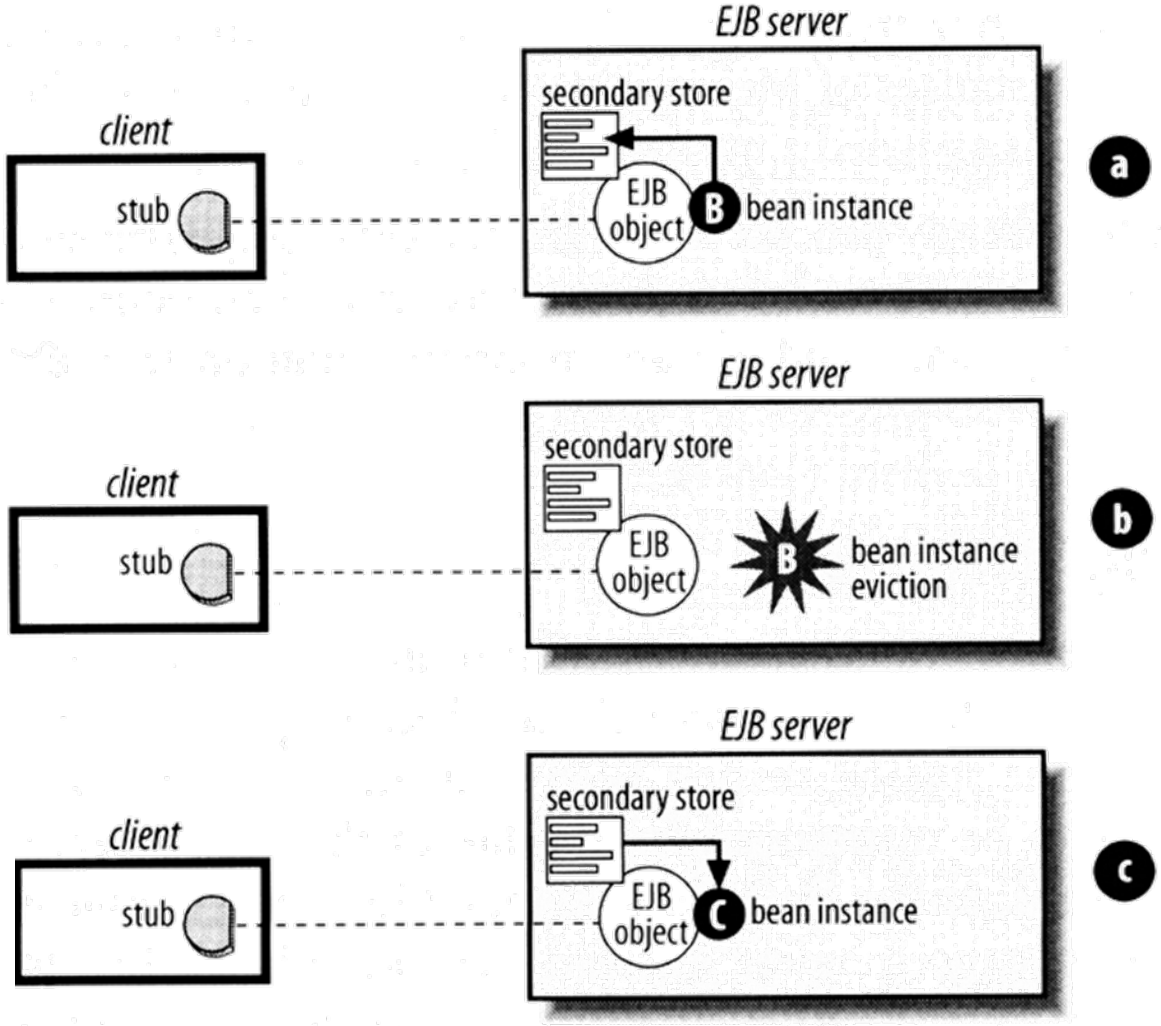


Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author)
Publisher: O'Reilly

Activación/“Passivation”

- Los **beans de sesión con estado** no participan en el “pooling” de instancias
  - El estado de la conversación con el cliente debe mantenerse durante toda la vida del servicio proporcionado a dicho cliente
- El contenedor usa activación/ “passivation”
  - para gestionar recursos
  - mecanismo transparente al cliente
- “Passivation”:
  - disociación del objeto EJB y la instancia del bean
  - serialización del estado de la instancia a almacenamiento secundario
- Activación:
  - Restauración del estado de la instancia relativa al objeto EJB

Proceso de Activación/“Passivation”

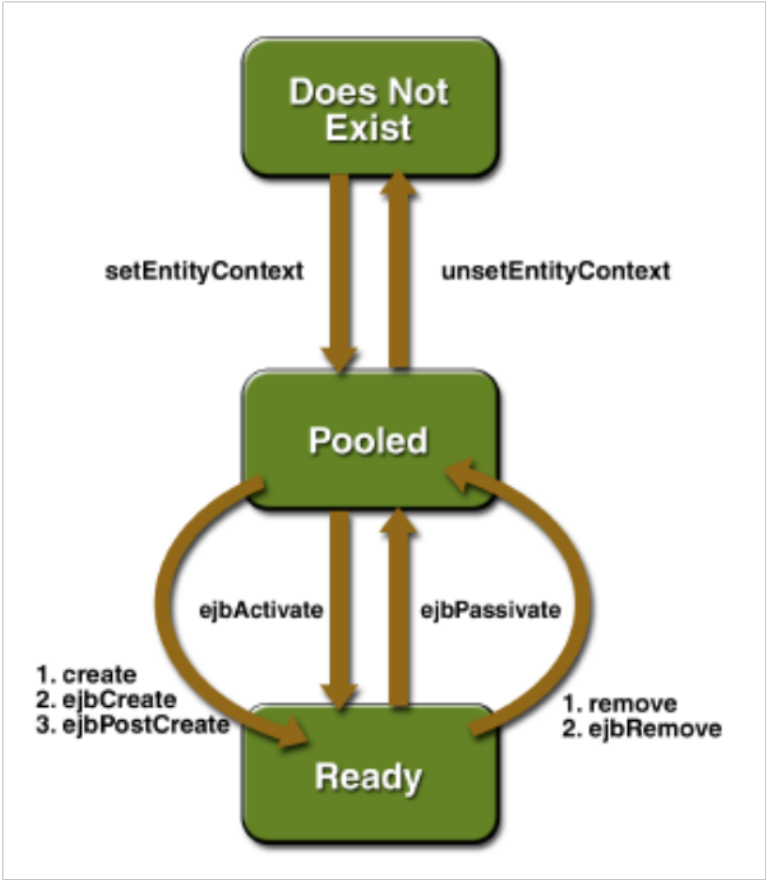


Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author)
Publisher: O’Reilly

Activación/“Passivation” Beans de entidad

- También aplica a los beans de entidad
  - para notificar a la instancia cuando va a ser “*swapped*” hacia o fuera del “pool” de instancia
- “Passivation”
  - Libera recursos
  - Almacena los datos que contiene en la base de datos subyacente
- Activación
  - Adquiere los recursos
  - Carga los datos de la base de datos subyacente

Ciclo de vida de un Entity Bean



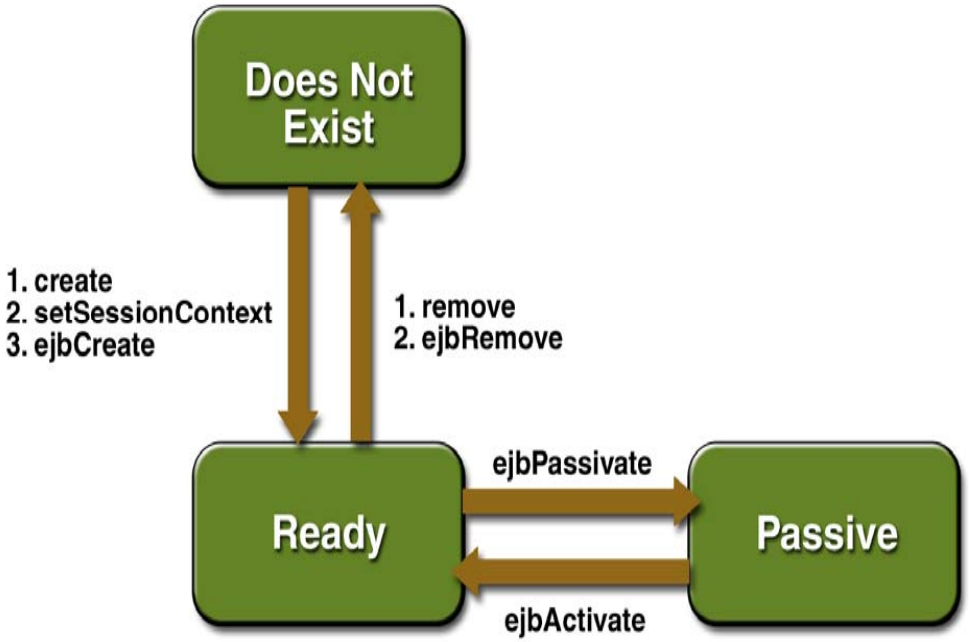
Fuente:
Java EE Tutorial 1.4 , Fourth Edition

API de un Entity Bean

Method Summary	
void	**ejbActivate()** A container invokes this method when the instance is taken out of the pool of available instances to become associated with a specific EJB object.
void	**ejbLoad()** A container invokes this method to instruct the instance to synchronize its state by loading it state from the underlying database.
void	**ejbPassivate()** A container invokes this method on an instance before the instance becomes disassociated with a specific EJB object.
void	**ejbRemove()** A container invokes this method before it removes the EJB object that is currently associated with the instance.
void	**ejbStore()** A container invokes this method to instruct the instance to synchronize its state by storing it to the underlying database.
void	**setEntityContext(**EntityContext ctx) Set the associated entity context.
void	**unsetEntityContext()** Unset the associated entity context.

Fuente:
http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/ejb/EntityBean.html

Ciclo de vida de los Session Beans



a) Statefull

b) Stateless

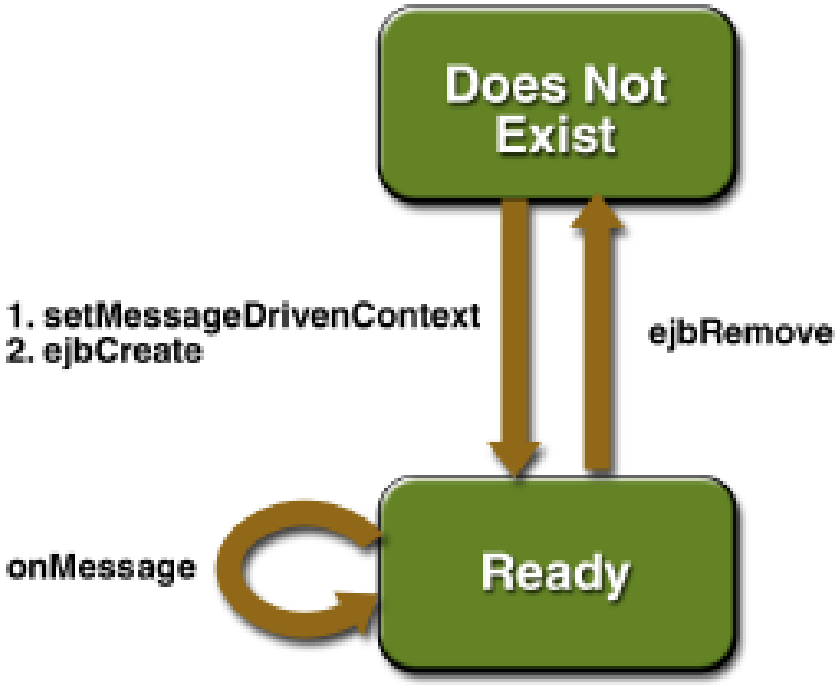
Fuente:
Java EE Tutorial 1.4 , Fourth Edition

API de los SessionBeans

Method Summary	
void	**ejbActivate()** The activate method is called when the instance is activated from its "passive" state.
void	**ejbPassivate()** The passivate method is called before the instance enters the "passive" state.
void	**ejbRemove()** A container invokes this method before it ends the life of the session object.
void	**setSessionContext(**SessionContext ctx) Set the associated session context.

Fuente:
http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/ejb/SessionBean.html

Ciclo de vida de un Message Bean



Fuente:
Java EE Tutorial 1.4 , Fourth Edition

API de un MessageBean

Method Summary	
void	**ejbRemove**() A container invokes this method before it ends the life of the message-driven object.
void	**setMessageDrivenContext**(MessageDrivenContext ctx) Set the associated message-driven context.

Fuente:
http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/ejb/MessageDrivenBean.html0

Persistencia

- Los beans de entidad son persistentes
  - Su estado se almacena permanentemente en una base de datos
  - Valores de atributos de la instancia del bean son sincronizados con la base de datos
- Persistencia manejada por contenedor (*Container-managed persistence,* CMP)
  - gestionada automáticamente
  - beans independiente de una base de datos especifica
  - el contenedor genera la lógica de acceso a la base de datos en tiempo de despliegue
- Persistencia manejada por bean (*Bean-managed persistence,* BMP)
  - gestionada manualmente (contenedor asistido)
  - desarrollador debe entender la estructura de la base de datos y APIs
  - más flexibilidad; por ejemplo, para sistemas legados no soportados por el vendedor
    - pero menos flexibilidad a la hora de realizar cambios

Implementaciones comunes de persistencia

- Persistencia objeto-relacional
  - el bean se representa como una tabla (o más)
    - Varias tablas: cambios en el bean de entidad requiere a menudo “joins” SQL
  - las propiedades (atributos) se representan como columnas de la tabla
  - cada instancia del bean será una fila de la tabla
  - CMP: el contenedor se encarga de mantener el estado de la instancia consistente con las tablas
    - **Sincronizar** el estado de la instancia del bean
  - CMP: intercepción en la creación y borrado de instancias
    - Resultados en la creación y borrado de registros de la BD
  - La identidad del bean de entidad representa un puntero a su estado en la BD

Implementaciones comunes de persistencia (II)

- Persistencia BD O-O
  - más limpia la asociación entre beans de entidad y la BD
  - más eficientes para grafos de objetos complejos
  - menos aceptadas y extendidas que las relacionales
- Persistencia legada
  - CMP requiere un contenedor especial
  - sistema legado no es soportado por el vendedor: BMP

Entity Bean de tipo CMP

- La clase de un bean de entidad CMP debe ser **abstracta**
- Los elementos persistentes se acceden mediante métodos especiales
  - No se declaran los atributos persistentes
  - Típicamente se utilizan getters y setters
    - `private abstract String getName();`
    - `private abstract String setName();`
  - El contenedor ha de generar el código necesario para acceder al sistema EIS (típicamente a la base de datos).
- No se suelen implementar los métodos de *callback*
  - No se implementan los métodos `ejbStore()`, `ejbLoad()`, `ejbActivate()`, `ejbPassivate()`, `setEntityContext()`, `unSetEntityContext()`, `ejbRemove()`.

Entity Bean de tipo CMP (II)

- Se suelen implementar de la interfaz métodos de tipo `create`
  - Cuando queremos poder crear nuevos objetos persistentes
  - Internamente hay un equivalente de tipo `ejbCreate`
- No se implementan los métodos de búsqueda
  - por cada método de búsqueda opcional (`findByXxx`)
    - habrá una entrada en el descriptor de despliegue, que utiliza EJB QL
  - **EJB QL ("*Query Language*")**
    - Una consulta EJB QL es una cadena que contiene tres cláusulas: SELECT, FROM, y opcionalmente WHERE
    - Por ejemplo:

```
<ejb-ql>
 SELECT OBJECT(p) FROM Persona AS p WHERE p.edad &gt; ?1
</ejb-ql>
```

- Los métodos de negocio
  - Cuando queremos poder crear nuevos objetos persistentes
  - Mantienen la signatura que se les da en la interfaz remota

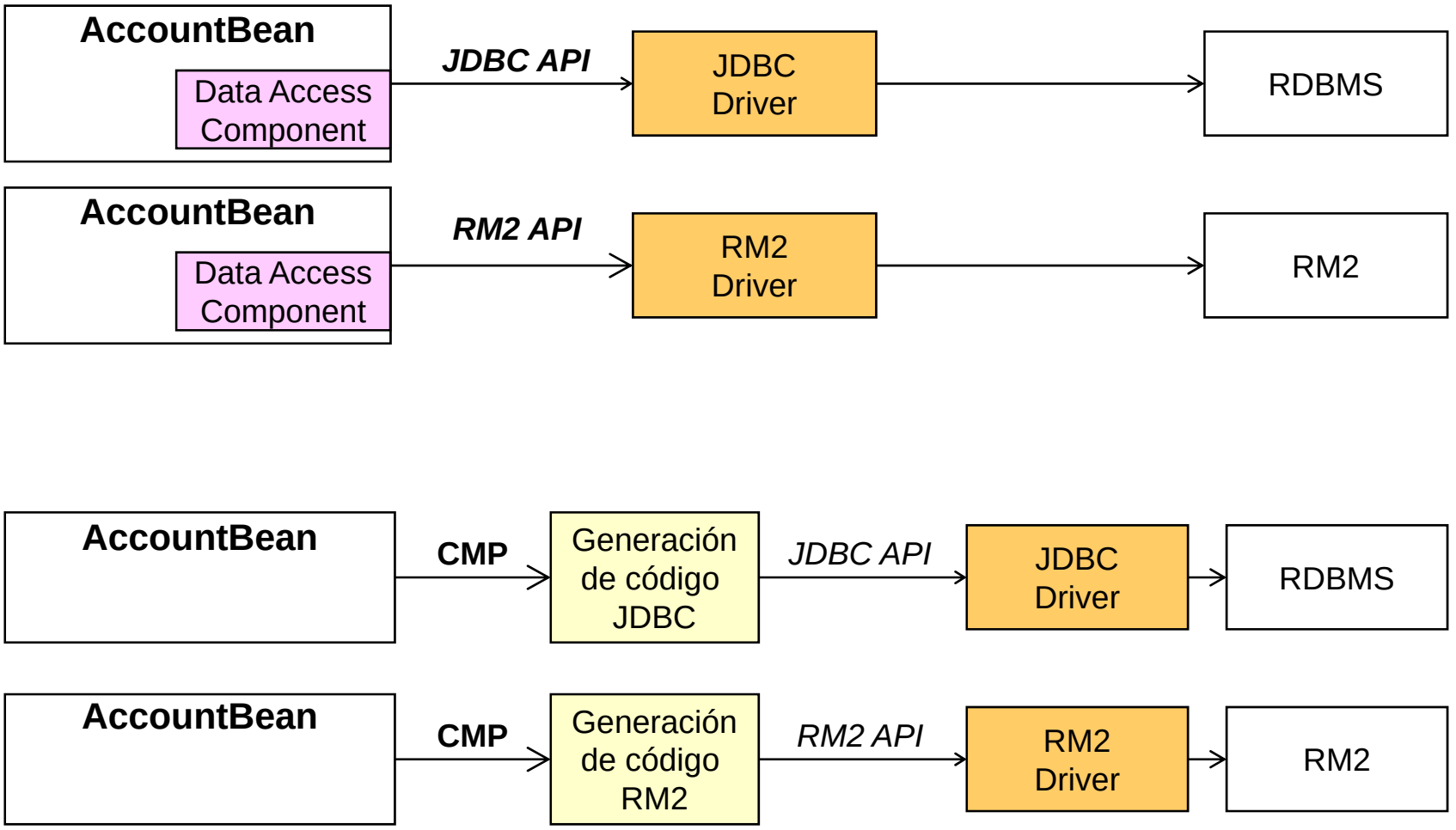
Entity Bean de tipo BMP

- Es una clase no abstracta
  - No hay métodos abstractos de tipo get ni set
- Se suelen implementar tres tipos de métodos
  - de creación, búsqueda y algunos de *callback*
    - `ejbCreate`
    - `ejbFindByPrimaryKey`
    - `ejbLoad`, `ejbStore`, `ejbRemove`, `setEntityContext`, `unsetEntityContext`
- El programador hay de realizar todas las comunicaciones con el EIS
  - Por ejemplo si es una base de datos
    - Se suele obtener debe obtener un recurso de conexión del ENC JNDI

```
Datasource ds =
(DataSource)cntx.lookup("java:comp/env/jdbc/titanDB");
```

    - Y también se crea el objeto data source
- En ellos el desarrollador es responsable de lanzar las excepciones en el momento correcto
  - Abre, cierra y reutiliza conexiones con la base de datos

Manejada por bean vs. por el contenedor



Ejemplo de componente Entity CMP

- Ejemplo bean de entidad: **`Cabin`**
  - Modela un camarote en un barco crucero
  - Interfaz remota: **`CabinRemote`**
  - Interfaz remota “home”: **`CabinHomeRemote`**
  - Clase: **`CabinBean`**

Interfaz remota: CabinRemote

```
package com.titan.cabin;
import java.rmi.RemoteException;

public interface CabinRemote extends javax.ejb.EJBObject {

  public String getName() throws RemoteException;
  public void setName(String str) throws RemoteException;
  public int getDeckLevel() throws RemoteException;
  public void setDeckLevel(int level) throws RemoteException;
  public int getShipId() throws RemoteException;
  public void setShipId(int sp) throws RemoteException;
  public int getBedCount() throws RemoteException;
  public void setBedCount(int bc) throws RemoteException;

}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Interfaz remota “home”: CabinHomeRemote

```
package com.titan.cabin;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.FinderException;

public interface CabinHomeRemote extends javax.ejb.EJBHome {


    public CabinRemote create(Integer id)
        throws CreateException, RemoteException;


    public CabinRemote findByPrimaryKey(Integer pk)
        throws FinderException, RemoteException;
}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O’Reilly

Clase bean: CabinBean (I)

```
package com.titan.cabin;

import javax.ejb.EntityContext;
import javax.ejb.CreateException;

public abstract class CabinBean
                      implements javax.ejb.EntityBean {

    public Integer ejbCreate(Integer id)
                                  throws CreateException {
              this.setId(id);
              return id;
    }

    public void ejbPostCreate(Integer id)
                                  throws CreateException {

    }

[…]
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Clase bean: CabinBean (II)

```
[…]
public abstract void setId(Integer id);
public abstract Integer getId();

public abstract void setShipId(int ship);
public abstract int getShipId( );

public abstract void setName(String name);
public abstract String getName( );

public abstract void setBedCount(int count);
public abstract int getBedCount( );

public abstract void setDeckLevel(int level);
public abstract int getDeckLevel( );

[…]
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Clase bean: CabinBean (III)

```
    public void setEntityContext(EntityContext ctx) {
        // Not implemented.
    }
    public void unsetEntityContext() {
        // Not implemented.
    }
    public void ejbActivate() {
        // Not implemented.
    }
    public void ejbPassivate() {
        // Not implemented.
    }
    public void ejbLoad() {
        // Not implemented.
    }
    public void ejbStore() {
        // Not implemented.
    }
    public void ejbRemove() {
        // Not implemented.
    }
}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Comentarios sobre el `CabinBean`

- El método **`ejbCreate`**
  - La interfaz home tiene un método **`create`** con la misma signatura
- El método **`findByPrimaryKey`**
  - La interfaz home define siempre este método
  - Los beans de entidad CMP no necesitan implementarlo (lo hace el contenedor)
  - "primary key" es algún objeto serializable
- Los atributos persistentes no se declaran explícitamente
  - tiene métodos abstractos modificadores y de acceso
- La interfaz remota no tiene un método `get` y `set` para el atributo `Id`
- No implementa ninguno de los métodos de *callback*
  - p. ej. `ejbStore, ejbLoad, ejbRemove`

Descriptor de despliegue CabinEJB

```
<!DOCTYPE ejb-jar PUBLIC "..." "http://...">
<ejb-jar>
<enterprise-beans>
   <entity>
     <ejb-name>CabinEJB</ejb-name>
     <home>com.titan.cabin.CabinHomeRemote</home>
     <remote>com.titan.cabin.CabinRemote</remote>
     <ejb-class>com.titan.cabin.CabinBean</ejb-class>
     <persistence-type>Container</persistence-type>
     <prim-key-class>java.lang.Integer</prim-key-class>
     <reentrant>False</reentrant>
     <abstract-schema-name>Cabin</abstract-schema-name>
                  . . .
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Descriptor de despliegue CabinEJB (II)

```
    <cmp-field><field-name>shipId</field-name></cmp-field>
    <cmp-field><field-name>name</field-name></cmp-field>
    <cmp-field><field-name>deckLevel</field-name></cmp-field>
    <cmp-field><field-name>bedCount</field-name></cmp-field>
    <cmp-field><field-name>id</field-name></cmp-field>
    <primkey-field>id</primkey-field>
    <security-identity><use-caller-identity />
     </security-identity>
   </entity>
</enterprise-beans>
<assembly-descriptor>
...
</assembly-descriptor>
</ejb-jar>
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Descriptor de despliegue CabinEJB (III)

```
. . .
<assembly-descriptor>
   <role-name>everyone</role-name>
   <method-permission>
        <method>
               <ejb-name>CabinEJB</ejb-name>
               <method-name>*</method-name>
        </method>
   </method-permission>
   <container-transaction>
        <method>
               <ejb-name>CabinEJB</ejb-name>
               <method-name>*</method-name>
        </method>
        <trans-attribute>Required</trans-attribute>
   </container-transaction>
</assembly-descriptor>
. . .
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Comentarios sobre el despliegue

- Se realiza con el fichero **`ejb-jar.xml`** (“*deployment descriptor*”) y con ayuda de
  - la herramienta de despliegue (“deploy tool”) del servidor EJB, o
  - Un entorno de desarrollo integrado (por ejemplo, NetBeans o Eclipse)

Creación de una tabla Cabin en la BD

- La herramienta de instalación/despliegue “mapea” beans de entidad a tablas de la bases de datos
  - Pero primero necesita instalar la BD y crear una tabla CABIN

- En este ejemplo, se utiliza la siguiente sentencia SQL:

```
create table CABIN
(
    ID int primary key NOT NULL,
    SHIP_ID int,
    BED_COUNT int,
    NAME char(30),
    DECK_LEVEL int
)
```

- En algunos casos se puede forzar a que el motor de el despliegue del EJB cree las tablas (opción: create tables on deployment)

Cliente remoto `Client1`

```
package com.titan.clients;

import com.titan.cabin.CabinHomeRemote;
import com.titan.cabin.CabinRemote;

import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import javax.rmi.PortableRemoteObject;
import java.rmi.RemoteException;
import java.util.Properties;
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Cliente remoto `Client1` (II)

```
public class Client1 {
  public static void main(String [] args) {
    try {
       Context jndiContext = getInitialContext();
       Object ref =
              jndiContext.lookup("CabinHomeRemote");
       CabinHomeRemote home = (CabinHomeRemote)
                  PortableRemoteObject.
                 narrow(ref,CabinHomeRemote.class);
       if (ref != null) {
          System.out.println("Found Cabin Home");
       }
       CabinRemote cabin1=home.create(new Integer(1));
       cabin1.setName("Master Suite");
       cabin1.setDeckLevel(1);
       cabin1.setShipId(1);
       cabin1.setBedCount(3);
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Cliente remoto `Client1` (III)

```
        Integer pk = new Integer(1);
        CabinRemote cabin2 = home.findByPrimaryKey(pk);
        System.out.println(cabin2.getName());
        System.out.println(cabin2.getDeckLevel());
        System.out.println(cabin2.getShipId());
        System.out.println(cabin2.getBedCount());

    } catch (java.rmi.RemoteException re) {
        re.printStackTrace();
    } catch (javax.naming.NamingException ne) {
        ne.printStackTrace();
    } catch (javax.ejb.CreateException ce) {
        ce.printStackTrace();
    } catch (javax.ejb.FinderException fe) {
        fe.printStackTrace();
    }
} // main
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Cliente remoto `Client1` (IV)

```
  public static Context getInitialContext()
                    throws javax.naming.NamingException {
    Properties p = new Properties();
    //… specify the JNDI Properties specific to the vendor
    p.put(Context.INITIAL_CONTEXT_FACTORY,
                "weblogic.jndi.WLInitialcontextFactory");
    p.put(Context.PROVIDER_URL, "t3://localhost:7001");
    return new javax.naming.InitialContext(p);
  } //getInitialcontext

} //class
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Pasos del cliente remoto

1. Obtener referencia del objeto home del bean
   1. Obtiene el contexto inicial JNDI (`InitialContext`)
   2. Ejecuta el método **`lookup()`** del home del bean en el contexto inicial
2. Hacer un "c*asting*" del resultado al objeto de tipo **`HomeRemote`**
   1. Ejecuta el método **`narrow()`** de la referencia obtenida desde el **`lookup()`**
      - Obtiene un stub que implementa la interfaz remota home
      - Necesario para compatibilidad CORBA (RMI sobre IIOP)
   2. Hace un "*casting*" del resultado (*Java native casting*) a la interfaz **`HomeRemote`**
3. Llamar al método **`create()`** sobre el objeto **`HomeRemote`**
   1. Crea la instancia del EJB (`CabinBean`)
   2. Crea el objeto EJB que envuelve la instancia del EJB
   3. Devuelve la referencia al objeto EJB (implementación de **`Remote`**)

Contexto inicial JNDI

- JNDI: API del servicio de nombres
  - Acceso uniforme a diferentes servicios de nombres
    - LDAP, NIS+, CORBA Naming,...
  - parecido a obtener una conexión al controlador de JDBC: acceso uniforme a diferentes bases de datos relacionales
- JNDI: organiza elementos EJB en estructura de directorios virtuales
  - beans, servicios, datos, recursos
    - vistos como directorios en el sistema de ficheros común
    - puede incluso estar distribuido
  - contexto inicial ≡ raíz del sistema de fichero JNDI
    - Objeto **`InitialContext`** creado en la llamada a **`getInitialContext()`**
- Objeto **`Properties`**
  - pasado al objeto **`InitialContext`** en la creación, el cual indica:
    - Dónde está el servidor EJB
    - Qué controlador JNDI (*JNDI service provider*) cargar
  - Parámetros son dependientes de la implementación

Contexto inicial en aplicaciones .ear

- Es más sencillo pues se puede revolver el problema haciendo uso de un método initial context

```
InitialContext ictx = new InitialContext();
Object h = ictx.lookup("java:comp/env/ejb/CabinEJB");
```

- De bastante utilidad cuando queremos acceder desde una pagina jsp a un entity almacenado dentro del mismo ear

"Lookup" en JNDI

- **`lookup()`** de la interfaz home
  - método de la clase **`javax.naming.Context`**
  - el nombre de objeto buscado es pasado como argumento
  - uso: `Object ref = jndiContext.lookup("CabinHomeRemote");`
- Bean como cliente de otro bean
  - bean puede buscar el home de otro bean en su propio JNDI ENC (*Environment Naming Context*)
  - ENC para cada bean creado en el despliegue:
    - ENC inicial: **`java:com/env`**
    - Localización de otros beans a los que accede: **`java:com/env/ejb`**
  - ejemplo: `Object ref = jndiContext.lookup("java:comp/env/ejb/CabinHomeRemote")`
- *lookup* de otros recursos: ENC da acceso a
  - las propiedades y otros recursos del contenedor
  - parámetros de inicialización, etc.

Narrow y Casting

- **`narrow()`**
  - método de la clase **`javax.rmi.PortableRemoteObject`**
    - definido en **`java.lang.Object`**
  - argumentos
    - referencia remota para ser “narrowed”
    - Tipo al cual debería ser “narrowed”
  - uso: `CabinHomeRemote home = (CabinHomeRemote) PortableRemoteObject.narrow(ref, CabinHomeRemote.class);`
- Especie de “casting” para objetos remotos
  - compatibilidad CORBA: RMI sobre IIOP
  - CORBA admite muchos lenguajes, no todos tienen un “casting” nativo
- Si la referencia devuelta tiene directamente el tipo adecuado
  - **`narrow()`** no es necesario
  - p.ej. **`create()`** devuelve directamente un objeto **`CabinRemote`**

Invocación de método remoto (RMI): Tipos

- Interfaces de componentes remotos utilizan tipos Java RMI
- Existen dos clases de tipos de retorno y parámetros
  - tipos declarados: verificados en tiempo de compilación
  - tipos reales: verificados en tiempo de ejecución
- Java RMI restringe los tipos de parámetros reales a:
  - Primitivos (`byte`, `boolean`, `char`, `int`, `long`, `double`, `float`)
  - String
  - remoto (**`java.rmi.Remote`**)
    - No tienen explícitamente que implementar **`java.rmi.Remote`**
  - "serializable" (**`java.io.Serializable`**)
    - No tienen explícitamente que implementar **`java.io.Serializable`**
    - p. ej. `java.util.Collection`
- La restricción no está en los tipos de parámetros declarados
  - Tipos declarados pueden no ser remotos / serializable

Tipos de parámetros “Serializable” y remoto

- “Serializable”:
  - se pasa por valor
  - se pasa una copia del objeto
  - cambios al objeto no son vistos por el objeto que tiene la copia

- Remoto:
  - se pasa como referencia remota
  - se pasa una copia del “stub” (paso por valor del stub)
  - la copia del stub apunta al mismo objeto remoto
  - cambios al objeto son vistos por todos los objetos que tienen referencias remotas

Serialización de referencias remotas: Handles

- Un “Handle” es
  - Una instancia de **`javax.ejb.Handle`**
  - Una referencia serializada al objeto EJB
- Un cliente puede
  - Serializar “handle” para almacenar la referencia al objeto remoto EJB
  - deserializar “handle” para recuperar la referencia al objeto remoto EJB
- Conversión a / desde un formato serializable
  - desde **`EJBObject`** a **`Handle`**: **`EJBObject.getHandle()`**
  - desde **`Handle`** a **`EJBObject`**: **`Handle.getEJBObject()`**
    - Debe ser “narrowed” a un tipo de interfaz remota adecuado

```
Handle handle = … // get Handle
Object ref = handle.getEJBObject();
CabinRemote cabin = (CabinRemote)
                                    PortableRemoteObject.narrow(...);
```

Beans de sesión con estado

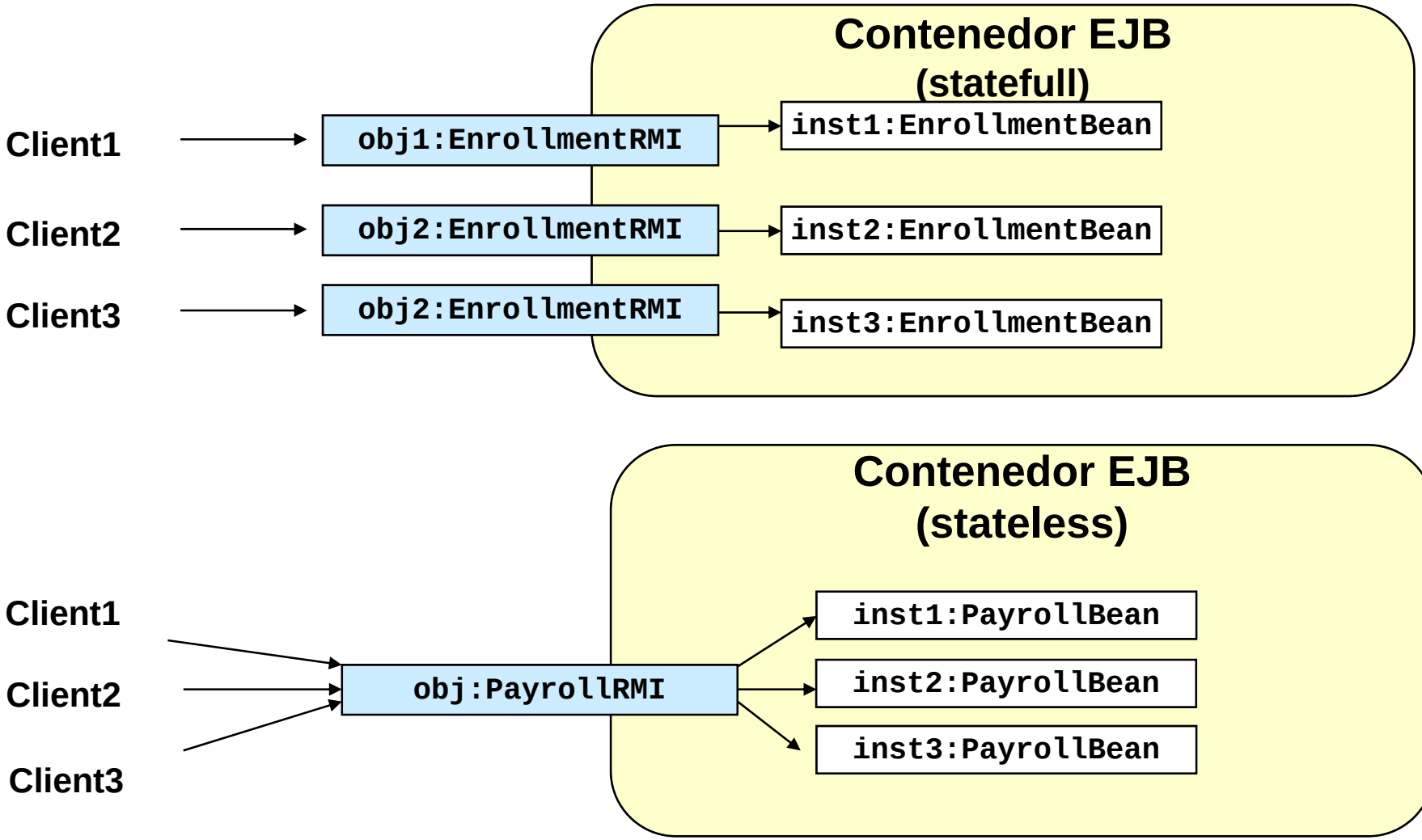
- Mantienen el estado del lado del servidor en representación de un solo cliente
- El estado conversacional consta de
  - variables de instancia definidas en la clase bean
    - datos leídos de la base datos (o un bean de entidad)
  - objetos alcanzables desde la variables de instancia
    - por ejemplo, datos introducidos por el usuario
- Si se utiliza desde un cliente servlet o JSP
  - la primera vez, el cliente obtiene la referencia del EJB
    - la almacena como atributo del objeto `HttpSession`
  - a continuación, para acceder al EJB se obtiene la referencia del objeto sesión
- No poseen métodos ni find ni unsetSessionContext

Bean de session sin estado (stateless)

- Desde el punto de vista del programador
  - Esta prohibido el guardar ningún tipo de estado dentro del entity bean
- Desde el punto de vista del contenedor
  - No hay diferencia entre clientes (puesto que no guardan ningún tipo de estado), lo cual permite que sean reutilizados por diferentes clientes
  - Se distingue de un session bean de tipo session de uno de tipo stateless mediante el atribution session-type

```
<session-type>Stateless</session-type>
<session-type>Stateful</session-type>
```

Statefull vs. Stateless



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

Ejemplo de Session Bean : TravelAgentEJB

- Ejemplo de un bean de sesión: `TravelAgent`
  - Modela un agente de viaje utilizado para reservar un viaje en un barco crucero
  - Interfaz remota: **`TravelAgentRemote`**
  - Interfaz remota home: **`TravelAgentHomeRemote`**
  - Clase: **`TravelAgentBean`**

Interfaz remota: TravelAgentRemote

```
package com.titan.travelagent;

import java.rmi.RemoteException;
import javax.ejb.FinderException;

public interface TravelAgentRemote
                          extends javax.ejb.EJBObject {

  // String elements follow the format
  // "id, name, deck level"
  public String [] listCabins(int shipID,
                   int bedCount) throws RemoteException;


}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Interfaz remota “home”: TravelAgentHomeRemote

```
package com.titan.travelagent;

import java.rmi.RemoteException;
import javax.ejb.CreateException;

public interface TravelAgentHomeRemote
                            extends javax.ejb.EJBHome {


    public TravelAgentRemote create()
                throws RemoteException, CreateException;

}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O’Reilly

Clase Bean: TravelAgentBean (I)

```
public class TravelAgentBean implements javax.ejb.SessionBean {

  public void ejbCreate() throws CreateException {
        // Do nothing.
  }

  public String [] listCabins(int shipID, int bedCount) {

    try {
      javax.naming.Context jndiContext = new InitialContext();
      Object obj =
      jndiContext.lookup("java:comp/env/ejb/CabinHomeRemote");

      CabinHomeRemote home = (CabinHomeRemote)
                             javax.rmi.PortableRemoteObject.
                             narrow(obj,CabinHomeRemote.class);
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Clase Bean: TravelAgentBean (II)

```
[...]
     Vector vect = new Vector();
     for (int i = 1; ; i++) {
        Integer pk = new Integer(i);
        CabinRemote cabin = null;
        try {
           cabin = home.findByPrimaryKey(pk);
        } catch(javax.ejb.FinderException fe) {
            break;
        }
        // Check to see if the bed count and ship ID match.
        if (cabin != null &&
          cabin.getShipId() == shipID &&
          cabin.getBedCount() == bedCount) {
          String details = i + "," + cabin.getName() + "," +
                           cabin.getDeckLevel();
          vect.addElement(details);
        }
     }
[...]
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Clase Bean: TravelAgentBean (III)

```
[...]
     String [] list = new String[vect.size()];
     vect.copyInto(list);
     return list;

    } catch(Exception e) {
        throw new EJBException(e);
    }
  }

  private javax.naming.Context getInitialContext()
                        throws javax.naming.NamingException {
     Properties p = new Properties();
     //... Specify the JNDI properties specific to the vendor.
     return new javax.naming.InitialContext(p);
  }

[...]
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Clase Bean: TravelAgentBean (IV)

```
[...]

  public void ejbRemove(){}
  public void ejbActivate(){}
  public void ejbPassivate(){}
  public void setSessionContext
                        (javax.ejb.SessionContext cntx){}

}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Descriptor de despliegue TravelAgentEJB

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise
   JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-jar_2_0.dtd">
<ejb-jar>
  <enterprise-beans>
   <session>
     <ejb-name>TravelAgentEJB</ejb-name>
     <home>com.titan.travelagent.TravelAgentHomeRemote</home>
     <remote>com.titan.travelagent.TravelAgentRemote</remote>
     <ejb-class>com.titan.travelagent.TravelAgentBean</ejb-class>
     <session-type>Stateless</session-type>
     <transaction-type>Container</transaction-type>
     <ejb-ref>
       <ejb-ref-name>ejb/CabinHomeRemote</ejb-ref-name>
       <ejb-ref-type>Entity</ejb-ref-type>
       <home>com.titan.cabin.CabinHomeRemote</home>
       <remote>com.titan.cabin.CabinRemote</remote>
     </ejb-ref>
     <security-identity><use-caller-identity/></security-identity>
   </session>
  </enterprise-beans>
  <assembly-descriptor> ... </assembly-descriptor>
</ejb-jar>
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Cliente TravelAgentClient (I)

```
public class TravelAgentClient{
   public static int SHIP_ID = 1;
   public static int BED_COUNT = 3;

   public static void main (String [] args) {
     try {
       Context jndiContext = getInitialContext();
       Object ref = jndiContext.lookup("TravelAgentHome");

       TravelAgentHomeRemote home = (TravelAgentHomeRemote)
              PortableRemoteObject.narrow(ref,
                                  TravelAgentHomeRemote.class);

       TravelAgentRemote travelAgent = home.create();
[...]
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Cliente de TravelAgentClient (II)

```
[...]
  //Get a list of all cabins on ship 1 with a bed count of 3.
        String list [] =
                    travelAgent.listCabins(SHIP_ID,BED_COUNT);
        for(int i = 0; i < list.length; i++){
            System.out.println(list[i]);
        }
      } catch(java.rmi.RemoteException re){
           re.printStackTrace();
      } catch(Throwable t){
           t.printStackTrace();
      }
      System.exit(0);
 }

 static public Context getInitialContext() throws Exception {
    Properties p = new Properties();
    //specify the JNDI properties specific to the vendor
    return new InitialContext(p);

  }
}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Ejercicio 1: Implementación de un Bean CMP

Crea un bean de entidad CMP que tenga las siguientes interfaces remotas: CustomerRemote y CustomerHomeRemote

```
package com.titan.customer;
public interface CustomerRemote
                                  extends javax.ejb.EJBObject {

  public String getLastName() throws RemoteException;
  public void setLastName(String lname)
                                   throws RemoteException;

  public String getFirstName() throws RemoteException;
  public void setFirstName(String fname)
                                    throws RemoteException;


}
```

```
public interface CustomerHomeRemote
                                   extends javax.ejb.EJBHome {

  public CustomerRemote create(Integer id)
                  throws CreateException, RemoteException;
  public CustomerRemote findByPrimaryKey(Integer id)
                  throws FinderException, RemoteException;


}
```

Ejercicio 2: Proceso de negocio soportado por un stateless

Cree un componente: IntercambiaUsuariosEJB, session bean de tipo stateless, que tenga un método (intercambia) que intercambie los datos de dos usuarios (`firstname` y `lastname`). Diseñe tanto la interfaces de la vista cliente como la implementación del bean.

```
public interface IntercambiaUsuarioRemote
                                    extends javax.ejb.EJBObject {

  public String intercambia(Integer user1, Integer user2) throws
RemoteException;

}
```

Fuente:
Enterprise JavaBeans, Fourth Edition
By Richard Monson-Haefel (Author), Bill Burke (Author), Sacha Labourey (Author) **Publisher:** O'Reilly

Cuestiones para profundizar (1/2)

Bloque I: Acciones llevadas a cabo por el servidor

- ¿Cuál es el principal motivo por el que se soporta pasivación en la arquitectura J2EE?
- El contenedor de EJBs, ¿ puede utilizar pooling con un Session de tipo stateful?
- En un SessionBean, ¿es posible acceder tomar acciones cuando este se pasivaza o activa?¿Cómo?

Bloque II: Ciclos de vida y API

- ¿Cuál es el principal motivo por el que se soporta pasivación en la arquitectura J2EE?
- ¿Para que sirven los métodos ejbActivate y ejbPassivate()?¿Por qué no aparecen en un Message Driven Bean?

Bloque III: Programación con CMP

- En la codificación de un Entity Bean de tipo CMP, ¿qué propiedad cumplen los métodos que permiten acceder al estado persistente del objeto?
- ¿Cuál es la diferencia entre entre ejbCreate y ejbPostCreate?
- Diga cuales son los pasos que da un cliente (de consola) al invocar a un EntityBean ya existente

Cuestiones para profundizar (2/2)

Bloque V: Creación de un Session Bean de tipo StateLess

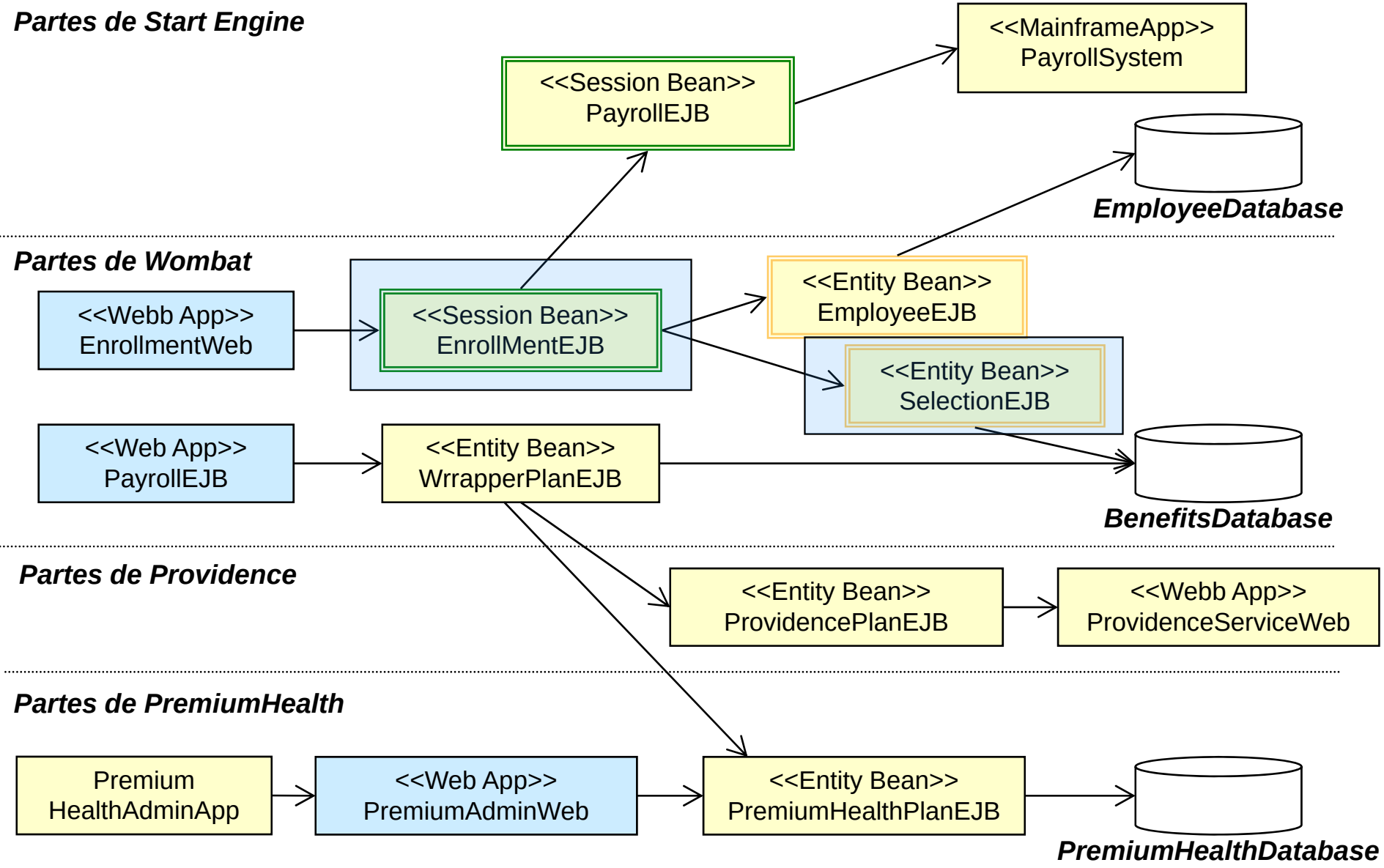
- ¿Cómo se sabe si un SessionBean es de tipo StateFull o StateLess?
- ¿Por qué un componente de tipo StateLess no tiene métodos find?

Material Auxiliar:

-Entity Bean tipo BMP (SelectionEJB)

-Session Bean de tipo Statefull (EnrollmentEJB)

Partes lógicas de la aplicación de Beneficios



SelectionEJB: Interfaz remota y Home

```
package com.wombat.benefits;
import javax.ejb.*;
import java.rmi.RemoteException;
import com.wombat.plan.Plan;
public interface Selection extends EJBObject{
      SelectionCopy getCopy()
            throws RemoteException, SelectionException;
      void updateFromCopy(Selection copy)
            throws RemoteException, SelectionException;
}
```

```
package com.wombat.benefits;

. . .
public interface SelectionHome extends EJBHome{
     Selection create(SelectionCopy copy)
       throws RemoteException, CreateException;
     Selection findByPrimaryKey(Integer emplNumber)
       throws RemoteException, FinderException;
     Selection findByEmployee(Employee employee)
       throws RemoteException, FinderException;
     Selection findByPlan(Plan plan)
       throws RemoteException, FinderException;
}
```

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

SelectionEJB: clases auxiliares

```
package com.wombat.benefits;
. . .
public interface SelectionCopy extends SelectionCopy{
     private Employee employee;
     private int coverage;
     private Plan medicalPlan;
     private boolean smokerStatus;

     public Employee getEmployee(){ . . .}
     public int getCoverage(){ . . .}
     public Plan getMedicalPlan(){ . . .}
     public boolean getSmokerStatus(){ . . .}

     public void getEmployee(Employee e){ . . .}
     public void getCoverage(int i){ . . .}
     public void getMedicalPlan(Plan p){ . . .}
     public void getSmokerStatus(boolean s){ . . .}
}
```

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

SelectionBean (I)

```
package com.wombat.benefits;

import javax.naming.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;
import javax.rmi.PortableRemoteObject.*;

import com.wombat.plan.*;

public class SelectionBean extends EntityBean{
     private EntityContext entityContext;
     private DataSource ds;

     //Métodos create
     public Integer ejbCreate(SelectionCopy copy)
       throws SelectionException, CreateException
     {
      try{
       Connection con =getConnection();
       PreparedStatement pstmt=con.prepareStatement(
            "INSERT INTO Selections"+
            "VALUES (?, ?,?,?)");
       pstmt.setInt(1, employeeNumber.intValue());
       pstmt.setInt(2, coverage);
       pstmt.setString(3, (String)medicalPlan.getPrimaryKey());
```

SelectionBean (II)

```
        pstmt.setString(3, (String)medicalPlan.getPrimaryKey());
        pstmt.setString(4, smokerStatus ? “Y”:N”);

        if(pstmt.executeUpdate()==1{
              con.close();
              return employeeNumber;
        }else{
              con.close();
              throw new CreateException();
        }

      }catch(SQLException ex){ throw new EJBException(ex);
      }catch(RemoteException ex){ throw new EJBException(ex);}  }

public Integer ejbFindByPrimaryKey(Integer employeeNumber)
      throws FinderException {
      try{
       Connection con = getConnection();
       PreparedStatement pstmt = con.prepareStatement(
              “SELECT sel_empl” +
              “FROM Selections” +
              “WHERE sel_empl = ?”);
       pstmt.setInt(1,employeeNumber.intValue());
       ResultSet rs=pstmt.executeQuery();
```

SelectionBean (III)

```
      if(rs.next()){
        con.close();
        throw new ObjectNotFoundException();
       }else {
        con.close();
        throw new ObjectNotFoundException();
       }
      }catch (SQLException ex){throw new EJBException(ex);}
  }
  public Integer ejbFindByPrimaryKey(Integer employeeNumber)
      throws  FinderException{
      try{
       Connection con =getConnection();
       PreparedStatement pstmt = con.prepareStatement(
              "SELECT sel_empl"+
              "FROM Selections"+
              "WHERE sel_empl=? ");
       pstmt.setInt(1,employeeNumber.intValue());
       ResultSet rs=pstmt.executeQuery();
       if(rs.next()){ con.close(); return employeeNumber;
       }else{ con.close(); throw new ObjectNotFoundException();}
      }catch(SQLException ex){throw new EJBException(ex);}
  }
```

SelectionBean (IV)

```
public Integer ejbFindByEmployee(Employee employee)
     throws FinderException{
     try{
      return ejbFindByPrimaryKey(
            (Integer)employee.getPrimaryKey());
     }catch(java.rmi.RemoteException ex){
       throw new EJBException(ex);
     }
}

public Collection ejbFindByPlan(Plan plan){
     try{
     PlanInfo planInfo = plan.getPlanInfo();
     int planType = planInfo.getPlanType();
     String planId = (String)planInfo.getPlanId();
     String columnName = PlanType.MEDICAL;

     Connection con = getConnection();
     PreparedStatement pstmt = con.prepareStatement(
            "SELECT sel_empl" +
            "FROM Selections" +
            "WHERE" +columnName+ " = ?" );
     pstmt.setString(1,planId);
     ResultSet rs=pstmt.executeQuery();
```

SelectionBean (V)

```
        Vector vec = new Vector();
        while (rs.next()){
         int emplnum=rs.getInt(1);
         vec.add(new Integer(emplnum));
        }
        con.close();
        return vec;
        }catch(Exception ex){throw new EJBException (ex);}
    }
    //Métodos de la interfaz EntityBean
    public void setEntityContext(EntityContext ctx){
        readEnvironment();
        this.ctx=ctx;}

    public void ejbRemove(){
        try{
         Connection con = getConnection();
         PreparedStatement pstmt = con.prepareStatement(
                "DELETE FROM Selections"+
                "WHERE sel_empl = ? ");
         pstmt.setInt(1, employeeNumber.invtValue());
         pstmt.executeUpdate();
         con.close();
        }catch(Exception ex){throw new EJBException(ex);}

    }
```

SelectionBean (VI)

```
public void  ejbLoad(){
     try{
      String medicalPlanId;
      employeeNumber = (Integer)entityContext.getPrimaryKey();
      employee = employeeHome.findByPrimaryKey(employeeNumber);

      Connection con = getConnection();
      PreparedStatement pstmt = con.prepareStatement(
            "SELECT sel_coverage, sel_smoker," +
             "sel_medical_plan"+
            "FROM Selections"+
            "WHERE sel_empl = ?");
      pstmt.setInt(1,employeeNumber.intValue());
      ResultSet rs = pstmt.executeQuery();
      if(rs.next()){
            coverage = rs.getInt(1);
            smokerStatus = rs.getString(2).equals("Y");
            medicalPlanId = rs.getString(3);
            con.close();
      } else {
            throw new NoSuchEntityException();
      }
     medicalPlan =planHome.findByPlanId(medicalPlanId);
     dentalPlan = planHome.FindbyPlanId(dentalPlanId);
```

SelectionBean (VII)

```
        }catch (Exception ex) {throw new EJBException(ex); }
    }
    public void ejbStore(){
        try{
         Connection con =getConnection();
         PreparedStatement pstmt = con.prepareStatement(
                "UPDATE Selections SET"+
                "sel_coverage = ?,"+
                "sel_medical_plan = ?,"+
                "sel_smoker = ?"+
                "WHERE sel_empl = ?" );
         pstmt.setInt(1,coverage);
         pstmt.setString(2, (String) medicalPlan.getPrimaryKey());
         pstmt.setString(2, smokerStatus ? "Y" : "N");
         pstmt.executeUpdate();
         con.close();
        }catch(Exception e){throw new EJBException(ex);}
    }
    //Métodos auxiliares
```

SelectionBean (VIII)

```
     //Métodos auxiliares
private Connection getConnection(){
     try{ return ds.getConnection();}catch (Exception e)
           { throw new EJBException(ex);}
}
private readEnviroment(){
     try{ Context ictx = new InitialContext();
     planHome = (PlanHome) PortableRemoteObject.narrow(
           ictx.lookup("java:comp/env/ejb/PlanEJB"),
           PlanHome.class);
     employeeHome =(EmployeeHome)
           PortableRemoteObject.narrow(
           ictx.lookup("java:comp/env/jdbc/EmployeeDB"),
           EmployeeHome.class;
     benefitsHome =(EmployeeHome)
           PortableRemoteObject.narrow(
           ictx.lookup("java:comp/env/jdbc/BenefitsDB"),
           BenefitsHome.class);
     }catch(Exception ex){  throw new EJBException(ex);   }
}
```

SelectionBean (IX)

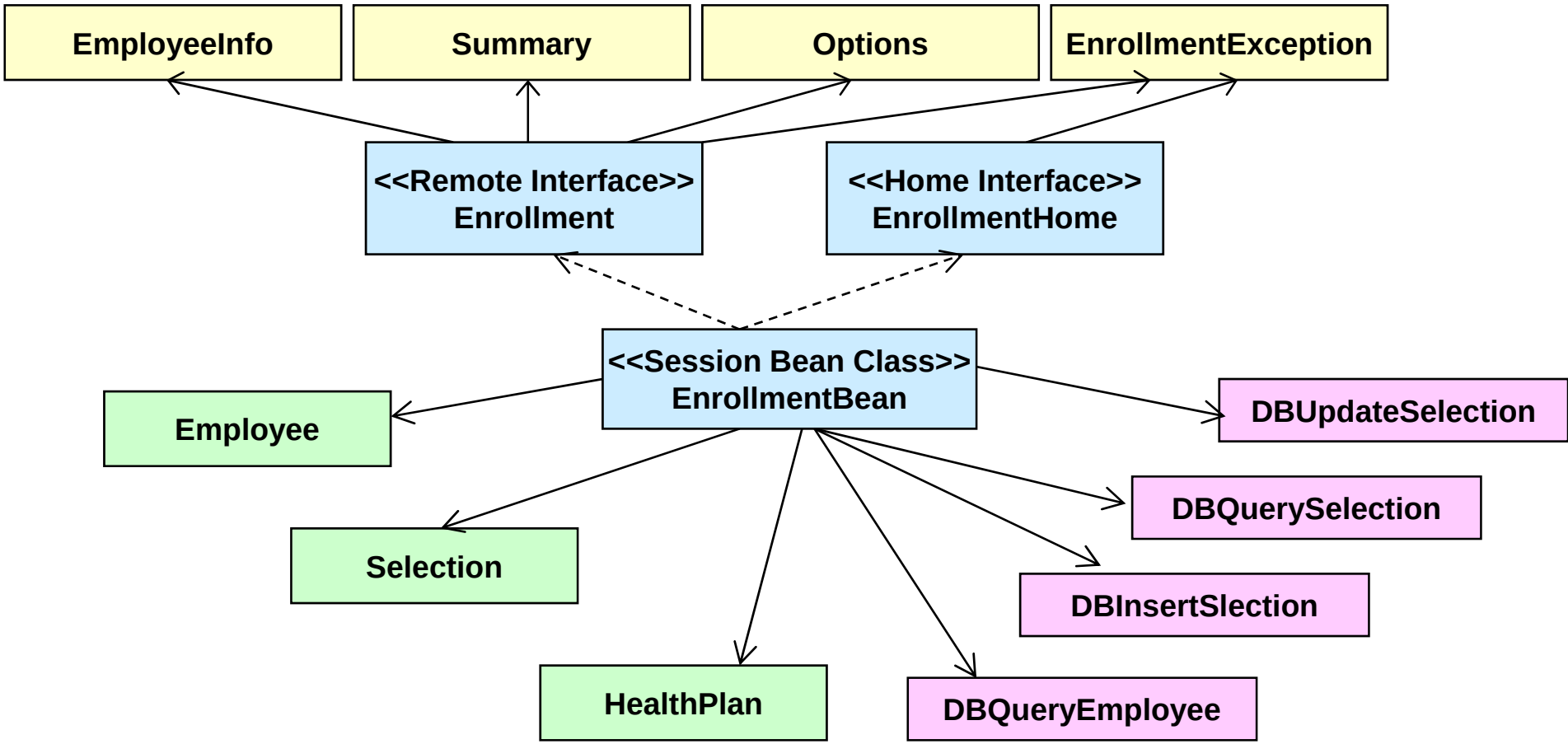
```
//Variables del estado
private int coverage;
private boolean smokerStatus;
private Employee employee;
private Plan medicalPlan;
private Integer employeeNumber;

//Variables internas
boolean checkPlanType;
...

//Métodos de negocio
public SelectionCopy getCopy(){
 SelectionCopy copy = new SelectionCopy();
 copy.setEmployee(employee);
 copy.setCoverage(coverage);
 copy.setMedicalPlan(medicalPlan);
 copy.setSmokerStatus(smokerStatus);
 return copy;}
```

SelectionBean (X)

```
        public void updateFromCopy(SelectionCopy copy)
         throws SelectionException{
          try{
           updateMedicalPlan(copy.getMedicalPlan());
           updateSmokerStatus(copy.getSmokerStatus());
           updateCoverage(copy.getCoverage());
          }catch(RemoteException ex){ throw new EJBException(ex);}
         }
        public void updateMedicalPlan(Plan p){medicalPlan =p}
        public void updateCoverage(int v){ coverage=v; }
        public void updateSmokerStatus(boolean v){
                        smokerStatus=v;}
} //@SelectionBean
```

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

Partes del EnrollmentEJB Session Bean



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

Enrollment Remote Interface (métodos de negocio)

```
package com.star.benefits;
import javax.ejb.*;
import java.rmi.RemoteException;
public interface Enrollment extends javax.ejb.EJBObject{

    EmployeeInfo getEmployeeInfo()
    throws RemoteException, EnrollmenteException;

    Options getCoverageOptions()
    throws RemoteException, EnrollmenteException;
    void setCoverageOptions(int choice)
    throws RemoteException,EnrollmenteException;

    Options getMedicalOptions()
    throws RemoteException, EnrollmenteException;
    void setMedicalOptions(int choice)
          throws RemoteException,EnrollmenteException;

    boolean getSmokerStatus()
          throws RemoteException, EnrollmenteException;
    void setSmokerStatus(boolean status)
    throws RemoteException,EnrollmenteException;
    Summary getSummary()
    throws RemoteException, EnrollmenteException;
    void commitSelections(boolean status)
    throws RemoteException,EnrollmenteException;

}//Enrollment remote interface
```

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

EnrollmentHome Home Interface (métodos de vida)

```
package com.star.benefits;
import javax.ejb.*;
import java.rmi.RemoteException;
public interface EnrollmentHome extends javax.ejb.EJBHome{

     Enrollment create(int emplnum)
     throws RemoteException, CreateException
     EnrollmenteException;


}//EnrollmentHome remote interface
```

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 las clases auxiliares (Home y Bean)

EmployeeInfo	Summary	Options	EnrollmentException

```
package com.star.benefits;
public class EmployeeInfo implements java.io.Serializable{
     int employeeNumber;
     String firstName;
     String lastName

public EmployeeInfo(){}

public EmployeeInfo (int emplnum, String fname, String lname){
     employeeNumber=emplnum; firstName=fname; lastname=lname;  }

public int getEmployeeNumber(){return employeeNumber;}
public String getFirstName(){return firstName;}
public int getLastName(){return LastNamer;}

public void setEmployeeNumber(int val){employeeNumber=val;}
public void setFirstName(String val){firstName=val;}
public void setLastName(String val){lastName=val;}
} //Employee Helper class
```

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

Esqueleto de la clase

```
package                                                com.star.benefits;
public class EnrollmentBean implements javax.ejb.SessionBean{
    public EnrollmentBean(){super();}

    //Implementación de los business methods
    public EmployeeInfo getEmployeeInfo(){. . .}
    public Options getCoverageOptions(){. . .}
    public void setCoverageOptions(int choice){. . .}
    public Options getMedicalOptions(){. . .}
    public void setMedicalOptions(int choice){. . .}
    public boolean getSmokerStatus(){. . .}
    public void setSmokerStatus(boolean status){. . .}
    public Summary getSummary(){. . .}
    public void commitSelections(){. . .}

    //Implementación de los métodos create(…)
    public EmployeeInfo ejbCreate(int emplNum){. . .}

    //Implementación de los javax.ejb.SessionBean
    public EmployeeInfo ejbRemove(){. . .}
    public EmployeeInfo ejbPassivate(){. . .}
    public EmployeeInfo ejbActivate(){. . .}
    public EmployeeInfo setSessionContext(SessionContext sc){...}

    //Métodos auxiliares
    private void calculateCostAndDeduction(){...}
    private void getDataSources(){...}

    } //Clase EnrollmentBean
```

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 EnrollmentBean (I)

```
package com.star.benefits;

        import javax.ejb;

        import javax.naming.Context;
        import javax.naming.InitialContext;
        import com.start.payroll.Payroll;
        import com.start.payroll.PayrollHome;
        import java.util.Date;
        import java.sql.SQLException;
        import java.sql.DataSource;
        import javax.rmi.PortableRemoteObject;

        import com.star.benifits.db.DBQueryEmployee;
        import com.star.benifits.db.DBQuerySelection;
        import com.star.benifits.db.DBInsertSelection;
        import com.star.benifits.db.DBUpdateSelection;

        //Representa la información del empleado

        class Employee{
                int emplNumber;
                String firstName;
                String lastName;
                Date birthDate;
                Date startDate;
                }
```

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 EnrollmentBean (II)

```
//Representa la selección
class Selection{
     int emplNumber;
     int coverage;
     String medicalplan;
     boolean smokerStatus;
     }
//La clase EnrollmentBean
public class EnroollmentBean implements SessionBean {
     private final static String[] coverageDescriptions = {
             “employee only”,
             “employee and espouse”,
             “employee, spouse and children” };
     //tablas internas al EJB
     private HealthPlan[] medicalPlans;
     private double employeeCostFactor=0.10;
     private int employeeNumber;
     private Selection selection;
     private boolean createSelection;
     private int age;       //Variable auxiliares
     private int medical selection =-1;
     private double totalCost;
     private double payrollDeduction
     private DataSource employeeDS; //JDBC data structures
     private DataSource benefitsDS;
```

Detalles de la clase EnrollmentBean (III)

```
private Payroll payroll;

//public no-arg constructor
public EnrollmentBean(){}

//Business methods follow.

//Get employee information.
public EmployeeInfo getEmployeeInfo(){
     return new EmployeeInfo(employeeNumber,
     employee.firstName, employee.lastName);

}

//Get Coverage options.
public Options getCoverageOptions(){
     Options opt= new Options(coverageDescriptions.length);
     opt.setOptionDescription(coverageDescriptions);
     opt.setSelectedOption(selection.coverage);
     return opt; }

//Set selected coverage option.
public void setCoverageOption(int choice)
     throws EnrollmentException{
     if (choice >=0 && choice<coverageDescriptions.length){
     selection.coverage=choice;
     } else { throw new EnrollmenteException(
               EnrollmenteException.INVAL_PARAM)};

     }
```

Detalles de la clase EnrollmentBean (IV)

```
//Get lists of available medical options
public Options getMedicalOptions(){
     Options opt=new Options(medicalPlans.length);
     for (int i=0; i< medicalPlans.length; i++){
             HealthPlan plan = medicalPlans[i];
             opt.setOptionDescription(i,plan.getDescription());
             opt.setOptionCost(i,
               plan.getCost(selection.coverage, age,
                            selection.smokerStatus));
             }
     opt.setSelectedOption(medicalSelection);
     return opt;
}
//Set selected medical option.
public void setMedicalOption(int choice)
     throws EnrollmentException{
     if(choice >=0 && choice<medicalPlans.length){
     medicalSelection=choice;
     selection.medicalPlanId=medicalPlans[choice].getPlanId();
     }     else     {     throw     new     EnrollmentException(
     EnrollmentException.INVAL_PARAM); }
}
```

Detalles de la clase EnrollmentBean (V)

```
//Get Smoker status.
public boolean getSmokerStatus(){
     return slection.smokerStatus;
}
//Set smoker status
public void setSmokerStatus(boolean status){
     selection.smokerStatus=status;
}
//Get summary of selected options and their cost.
public Summary getSummary(){
     calculateTotalCostAndPayrollDeductions();
     Summary s=new Summary();
     s.setCoverageDescription(
            coverageDescription[selection.coverage]);
     s.setSmokerStatus(selection.smokerStatus);
     s.setMedicalDescription(
            medicalPlans[medicalSelection].getCost(
                   selection.coverage, age,
                   selection.smokerStatus));
     s.setTotalCost(totalCost);
     s.setPayrollDeduction(payrollDeduction);
     return s;
}
```

Detalles de la clase EnrollmentBean (VI)

```
//Update corporate databases with the new selections.
public void commitSelections(){
     if(createSelection){
     DBInsertSelection cmd1 = null;
     try{
      cmd1=new DBInsertSelection(benefitsDS);
      cmd1.setEmplNumber(employeeNumber);
      cmd1.setCoverage(selection.coverage);
      cmd1.setMedicalPlanId(selection.medicalPlanId);
      cmd1.setSmokerStatus(selection.smokerStatus);
      cmd1.execute();
      createSelection =false; }catch (SQLException ex) {
                              throw new EJBException(ex);} finally
            { if (cmd1 !=null) cmd1.release();}
     } else {
     DBUpdateSelection cmd2 =null;
     try{
      cmd2 =new DBUpdateSelection(benefitsDS);
      cmd2.setEmplNumber(employeeNumber);
      cmd2.setCoverage(selection.coverage);
      cmd2.setMedicalPlanID(selection.medicalPlanID);
      cmd2.setSmokerStatus(selection.smokerStatus);
      cmd2.execute(); }catch(SQLException ex){
                    throw new EJBException(ex);} finally{
                    if (cmd2 !=null) cmd.release();}
     }
```

Detalles de la clase EnrollmentBean (VII)

```
//Update information in the payroll system
try{
     payroll.setBenefitsDeduction(employeeNumber,
            payrollDeduction);
     }catch(Exception ex){
            throw new EJBException(ex);
     }
}/@commitSelections

//Life methods (create,  ejbRemove, ejbPasivate and activate)
public void ejbCreate(int emplNum) throws
     EnrollmentException{
     employeeNumber= emplNum;
     //Obtaining the environment entries
     readEnviromentEntries();
     //Obtain JDBC data sources from the environment
     getDataSources();
     //Read employee infomration.
     DBQueryEmployee cmd1=null;
     try{
      cmd1 =new DBQueryEmployee(employeeDS);
      cmd1.setEmployeeNumber(emplNum);
      cmd1.execute();
      if (cmd1.next()){
            employee =new Employee();
```

Detalles de la clase EnrollmentBean (VIII)

```
                employee.emplNumber =emplNum;
                employee.firstName = cmd1.getFirstName();
                employee.lastName  =cmd1.getLastName();
                employee.startDate = cmd1.getStartDate();
                employee.birthDate = cmd1.getBirthDate();
        } else {
                throw new EnrollmentException
                        ("no employee record");
                }
    }catch (SQLException ex){
        throw new EJBException(ex);
    }finally{
        if (cmd1 !=null) cmd1.release();
        }
    //Read the previous benefits selection
    DBQuerySelection cmd2=null;
        try{
         cmd2 = new DBQuerySelection(benefitsDS);
         cmd2.setEmployeeNumber(emplNum);
         cmd2.execute();
         if (cmd2.next()){
                selection = new Selection();
                selection.emplNumber= emplNum;
                selection.coverage= cmd2.getCoverage();
                selection.medicalPlanID = cmd2.getMedicalPlanId();
```

Detalles de la clase EnrollmentBean (IX)

```
                selection.smokerStatus= cmd2.getSmokerStatus();
                createSelection = false;
        }else{
         //No previous selection
         selection=new Selection();
         selection.emplNumber= emplNum;
         selection.coverage=0;
         selection.medicalPlanID = medicalPlans[0].getPlanId();
         selection.smokerStatus =false;
         createSelection = true;
        }
}catch (SQLException ex){
        throw new EJBException(ex);
} finally {
        if (cmd2 !=null) cmd2.release();}
//Calculate employee's age.
java.util.Date today = new java.util.Date();
age = (int ((today.getTime()-employee.birthDate.getTime()) /
((long) 365*24*60*60*1000));
//Translate the dical plan ID to an index
for (int i=0; i<medicalPlans.length; i++){
        if (medicalPlans[i].getPlanId().equals(
            selection.medicalPlanId)){
            medicalSelection=i;
            break;                    }
        }
}//@ejbCreate
```

Detalles de la clase EnrollmentBean (X)

```
//Clean up any resource held by the instance
public void ejbRemove(){
     try{
      payroll.remove();
     }catch(Exception ex){
     }

}
//Release state that cannot be preserved across passivation
public void ejbPassivate(){
     employeeDS =null;
     benefitsDS =null;

}
//Reacquire state released before passivation
public void ejbActivate(){
     getDataSources();

}
//Session context
public void setSessionContext(SessionContext sc) {}
```

Detalles de la clase EnrollmentBean (XI)

```
//Helper methods follow.

//Calculate total benefits cost and payroll deduction.
private void calculateTotalCostAndPayrollDeductions(){
     double medicalCost =
     medicalPlans[medicalSelection].getCost(
             selection.coverage, age, selection.smokerStatus);
     totalCost=medicalCost;
     payrollDeduction=totalCost*employeeCostFactor;
     }

//Read and process enterprise bean's environment entries.
private void readEnvironmentEntries(){
try{
     Context ictx = new InitialContext();
     String medicalPlanList = (String)
            ictx.lookup("java:comp/env/medicalPlans");
     String[] medicalPlanClassNames = parseClassNames(
         medicalPlanList);
     medicalPlans = new HealthPlan[medicalPlanClassNames.length];
     for(int i=0; i<medicalPlanClassNames.length; i++){
       medicalPlans[i]=(HealthPlan)Class.forName(
            medicalPlanClassNames[i]).newInstance();
     }
```

Detalles de la clase EnrollmentBean (XII)

```
        PayrollHome payrollHome = (PayrollHome)
                PortableRemoteObject.narrow(
                 ictx.lookup("java.comp/env/ejb/PayrollEJB"),
                 PayrollHOme.class);
        payroll= (Payroll)payrollHome.create();
    } catch (Exception ex){
        ex.printStackTrace();
        throw new EJBException(ex);
        }
    }
    private getDataSources(){
        try{
        Context ictx = new InitialContext();
        employeeDS = (DataSource)ictx.lookup(
                "java:comp/env/jdbc/EmployeeDB");
        benefitsDS = (DataSources)ictx.lookup(
                "java:comp/env/jdbc/BenefitsDB");
        } catch (Exception ex){
          ex.printStackTrace();
          throw new EJBException(ex);
        }
    }
```

Detalles de la clase EnrollmentBean (XIII)

```
   //Parse: separated class names.
   private static String[] parseClassNames(String list){
        String[] rv = new String[0];
        while (list.length() !=0){
         int x = list.indexOf(':');
         String name;
         if(x<0){
                name = list;
                list = "";
         } else {
                name = list.substring(0,x);
                list = list.substring(x+1);
         }
         if (name.length()==0){
                continue;
         }
         String[] orv =rv;
         rv = new String[rv.length+1];
         for (int i=0; i<orv.length; i++)
                rv[i]=orv[i];
         rv[rv.length -1]=name;
        }
        return rv;
   }
}//@EnrollmentBean
```

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

Interfaz HealthPlan e Insurance Plan Class

```
public interface HealthPlan {
     String getPlainId();
     String getDescription();
     double getCost(int coverage, int age, boolean smokerStatus);
}
```

```
package com.star.plans;
import com.star.benefits.HeathPlan;
public class PremiumHealPOOPlan implements HealthPlan{
     public PremiumHealthPPOPlan(){ super();}
     public String getPlanId(){ return “PHPPO”;}
     public String getDescription(){ return “PremiumHealth PPO”;}
     public double getCost(int coverage, int age, boolean smoker){
          //Calculate the insurance premium based on the cvg
          return premium;
          }
}
```

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

DBQueryEmployee Command Bean

```
package com.star.benefits.db;

import java.sql.SQLException;
import java.util.Date;
import java.sql.DataSource;

public class DBQueryEmployee extends DBQueryBean {
     static String statement =
      "SELECT empl_first_name, empl_last_name, empl_birth_date"
       +"empl_start_date, empl_dept_id" +
      "FROM employees WHERE empl_id = ?";
     public  DBEmployee(DataSource  ds)  throws  SQLException{
      super(ds,statement); }
     public   void   setEmployeeNumber(int   emplNum)   throws
     SQLException{ pstmt.setInt(1,emplNum); }
     public String getFirstName() throws SQLException{
      return resultSet.getString(1);}
     public String getLastNameName() throws SQLException{
      return resultSet.getString(2);}
     public String getBirthDate() throws SQLException{
      return resultSet.getString(3);}
     public String getStartDate() throws SQLException{
      return resultSet.getString(4);}
     public  String  getDepartmentNumber()  throws  SQLException{
      return resultSet.getString(5);}

 }
```

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

DBUpdateSelection Command Bean

```
package com.star.benefits.db;

import java.sql.SQLException;
import java.sql.DataSource;

public class DBUpdateSelection extends DBUpdateBean {
     static String statement =
     "UPDATE Selections SET "+ "sel_coverage=?" +
     "sel_medical_plan" + "sel_smoker = ?" +
     "WHERE sel_empl = ?";
     public DBUpdateSelection(DataSource ds)
     throws SQLException{   super(ds,statement);  }
     public void setEmplNumber(int emplNum) throws
     SQLException{ pstmt.setInt(5,emplNum); }
     public void setCoverage(int cov) throws SQLException{
      pstmt.setInt(1,cov);}
     public void setMedicalPlanId(String id) throws
     SQLException{  pstm.setString(2,id);}                }
     public void setSmokerStatus(boolean st) throws
     SQLException{  pstmt.setString(3, st ? "Y":"N");}

 }
```

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

Deployment descriptor: statefull o stateless

```
...
<enterprise-bean>
     <session>
          <display-name> Enrollmente Bean</display-name>
          <ejb-name>EnrollmentEJB</ejb-name>
          <home>com.start.benefits.EnrollmentHome</home>
          <remote>com.start.benefits.Enrollment</remote>
          <session-type>Statefull</session-type>
     </session>
<enterprise-bean>

...
<enterprise-bean>
     <session>
          <display-name>Payroll Bean</display-name>
          <ejb-name>PayrollEJB</ejb-name>
          <home>com.start.benefits.PayrollHome</home>
          <remote>com.start.benefits.Payroll</remote>
          <session-type>Stateless</session-type>
     </session>
</enterprise-bean>
```

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