ISIL APRENDE HACIENDO

TITULO

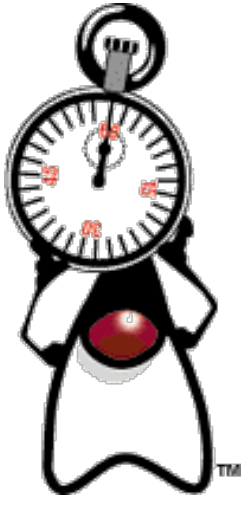
APLICACIONES EMPRESARIALES II

Semana Nro 02

Arquitectura de EJBs

Agenda

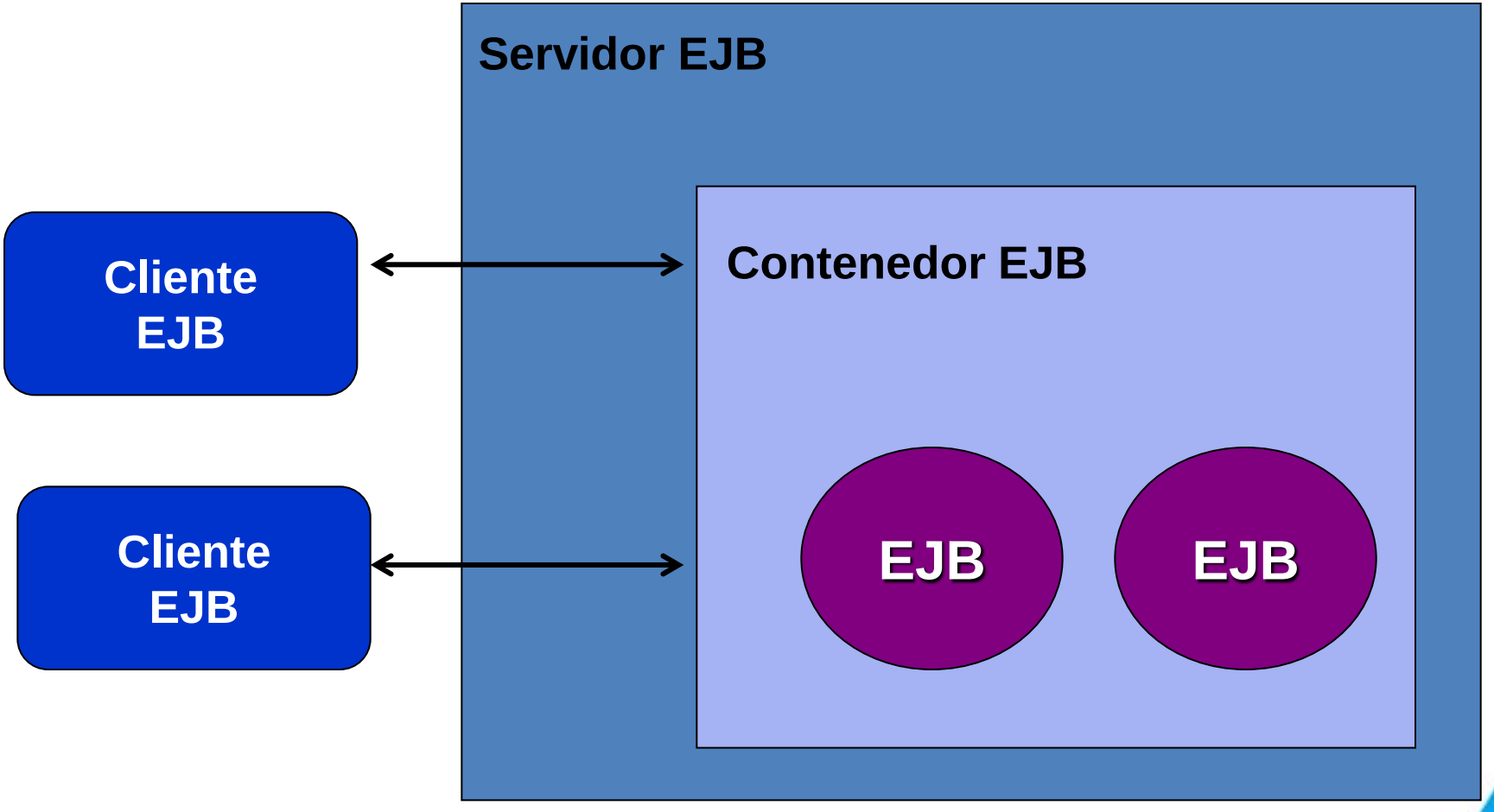
- Arquitectura EJB.
- Como trabajan los EJBs.
- EJB Remoto.
- EJB Local.
- Extras para el desarrollo de EJBs.



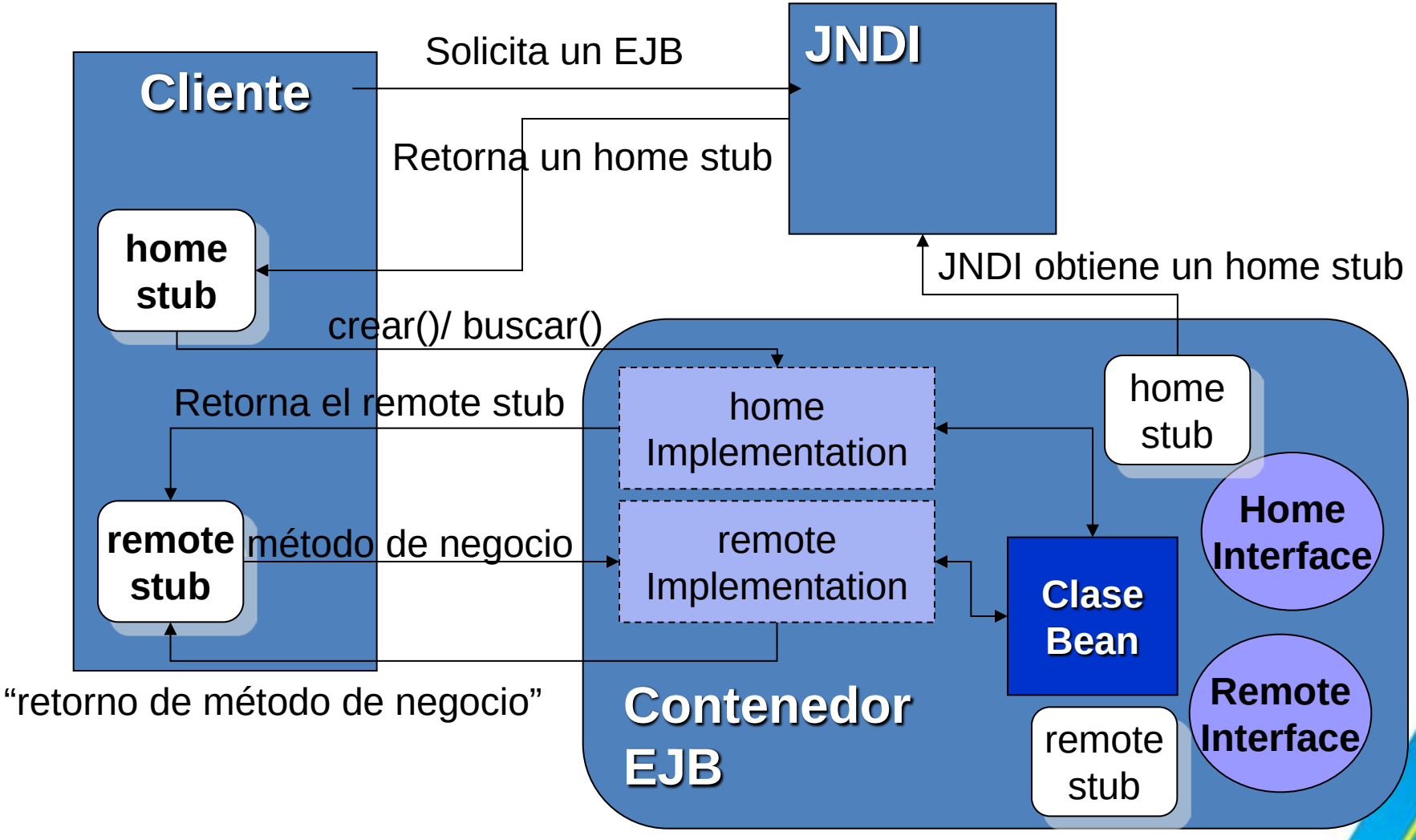
Arquitectura EJB

- Un Servidor de EJB (WebLogic).
- Contenedores EJB.
- Clientes EJB.
- Enterprise Java Beans.
- Servicio de Nombramiento y Directorios (JNDI)

Arquitectura EJB



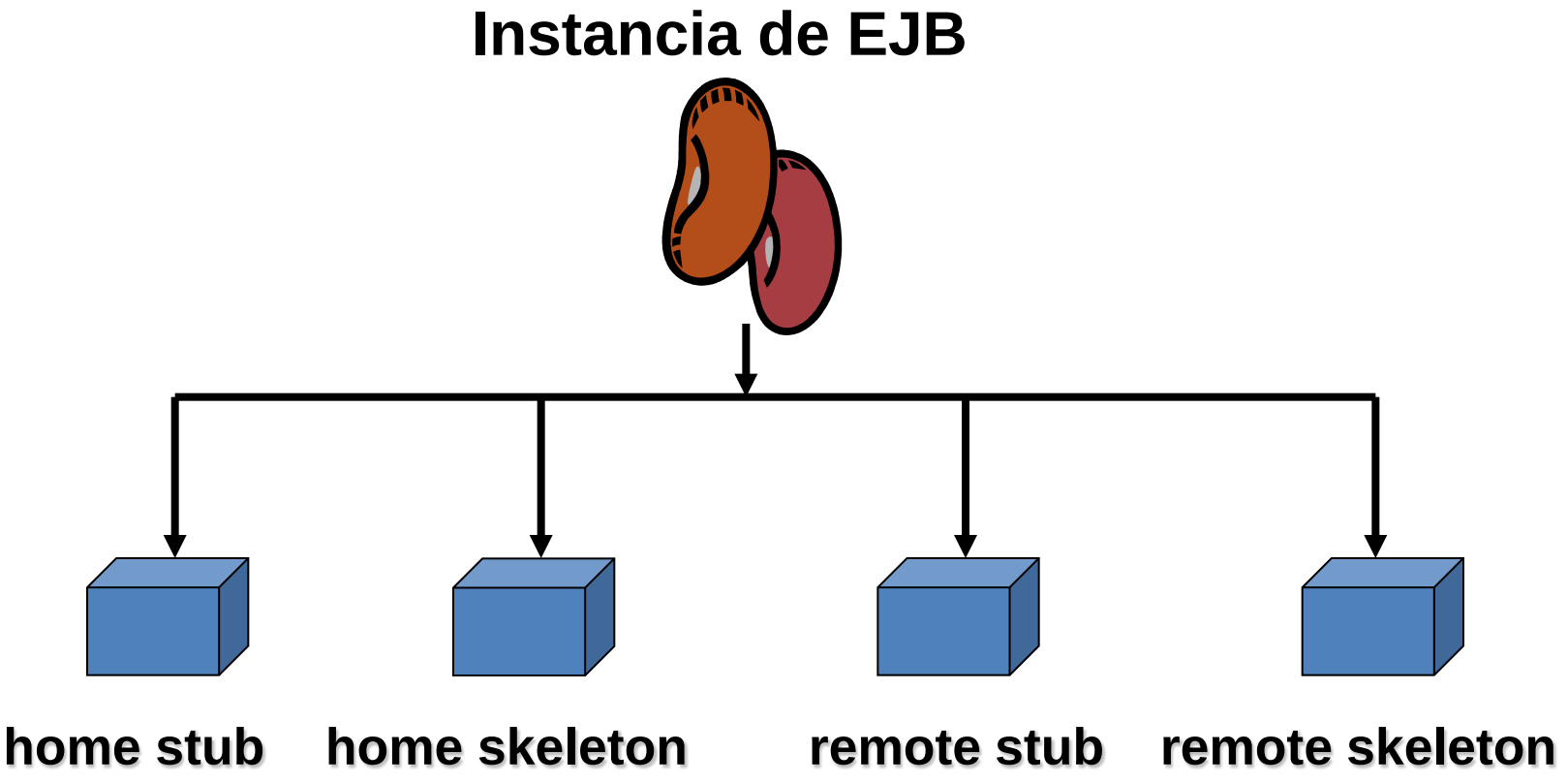
Como trabajan los EJBs



Local Interface

- Permite by pasear la comunicación RMI, de tal manera que el cliente pueda acceder al EJB de manera local.
- Se minimiza la sobrecarga de memoria en el lado del servidor.

Home Interfase / Remote Interface



Objetos Home y Remote

- Home Interface
  - Responsable de manejar el ciclo de vida de un EJB.
  - Provee meta información de un EJB.
  - Es compartido por todos los clientes del EJB.
  - Tiene un stub ubicado dentro del JNDI.
- Remote Interface
  - Lista de operaciones de negocio de un EJB

EJB Remoto

Remote Interface

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

    public java.lang.String foo(  ) throws java.rmi.RemoteException;

    public java.lang.String mensajeRemoto( java.lang.String param )
        throws java.rmi.RemoteException;

}
```

Home Interface

```
public interface ClienteHome extends javax.ejb.EJBHome {
   public static final String COMP_NAME="java:comp/env/ejb/Cliente";
   public static final String JNDI_NAME="ejb/Remoto";

   public pe.com.sunat.ejb.Cliente create()
      throws javax.ejb.CreateException,java.rmi.RemoteException;

}
```

EJB Local

Local Home Interface

```
public interface ClienteLocalHome extends javax.ejb.EJBLocalHome {
   public static final String COMP_NAME="java:comp/env/ejb/ClienteLocal";
   public static final String JNDI_NAME="ejb/Local";

   public pe.com.sunat.ejb.ClienteLocal create()
      throws javax.ejb.CreateException;

}
```

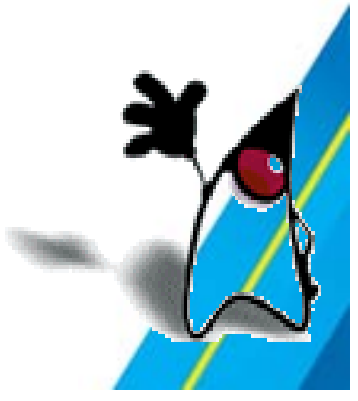
Local Interface

```
public interface ClienteLocal extends javax.ejb.EJBLocalObject {

        public java.lang.String mensajeLocal( java.lang.String param) ;

}
```

Session Beans

Edwin Maraví

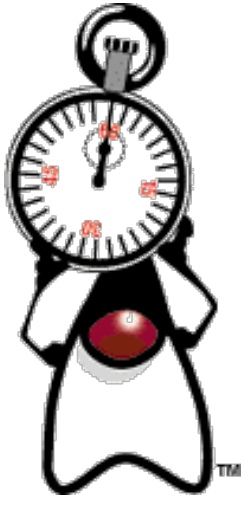
emaravi@cjavaperu.com



Agenda

En esta lección, usted aprenderá a:

- Describir las características de los beans de sesión
- Identificar los diferentes tipos de beans de sesión
- Explicar el ciclo de vida de los beans de sesión sin estado
- Crear y desplegar beans de sesión sin estado
- Crear aplicaciones utilizando beans de sesión sin estado



Preguntas de Evaluación

1. **¿Qué componente de los enterprise bean interactúa con el cliente, en nombre del Contenedor EJB, para proveer servicios, como control de transacciones, seguridad y persistencia?**
   a. Clase Enterprise Bean
   b. Interfaz Remota
   c. Interfaz Home
   d. Objeto EJB

2. **¿Quién tiene la responsabilidad de empaquetar los archivos de clase Java en archivos ejb-jar en la especificación EJB?**
   a. El proveedor del Bean
   b. El ensamblador de la aplicación
   c. El proveedor del contenedor EJB
   d. El proveedor del servidor EJB

Preguntas de Evaluación

3. **¿Quién tiene la responsabilidad en la especificación EJB de analizar un problema de negocio y ensamblar los componentes EJB de acuerdo con esto para resolver el problema?**
   a. Desplegador EJB
   b. Proveedor del servidor EJB
   c. Proveedor del Bean
   d. Ensamblador de la aplicación

4. **¿Qué bean habilita al contenedor EJB para gestionar las consultas a la base de datos y los asuntos de conectividad?**
   a. Message-driven bean
   b. Bean de entidad BMP
   c. Bean de entidad CMP
   d. Bean de sesión

Preguntas de Evaluación

5. **¿Qué bean implementa la lógica empresarial que requiere mensajería asíncrona entre los componentes de una aplicación EJB?**
   a. Bean conducido por objetos
   b. Bean de entidad BMP
   c. Bean de entidad CMP
   d. Bean de sesión

Información General de los Beans de Sesión

Los beans de sesión:

- Se utilizan para realizar las funciones de negocio de las aplicaciones empresariales.
- Tienen las siguientes características:
  - No son permanentes en naturaleza.
  - Implementan el estado conversacional entre un cliente y el servidor J2EE.
  - Dan servicio a una sola solicitud del cliente a la vez.
  - Le permiten acceder a los datos almacenados en una base de datos.
- Tienen los siguientes tipos:
  - **Beans de sesión sin estado**
  - **Beans de sesión con estado**

Información General de los Beans de Sesión

Características de un Bean de Sesión sin Estado:

- Ejecuta operaciones de negocio sin mantener el estado del cliente.
- Almacena el estado del cliente en las variables de la instancia solamente durante el tiempo en el cual el bean de sesión sin estado ejecuta un método.
- Maneja una solicitud del cliente procesada por un solo método del bean de sesión.
- Es un componente ligero. Un bean de sesión sin estado no contiene estructuras de datos complejas porque no almacena el estado del cliente.
- Contiene métodos que realizan funciones de negocio, las cuales no son específicas del cliente.
- Mejora la escalabilidad de una aplicación porque un menor número de instancias del bean de sesión sin estado pueden dar servicios a un mayor número de solicitudes del cliente.

Información General de los Beans de Sesión

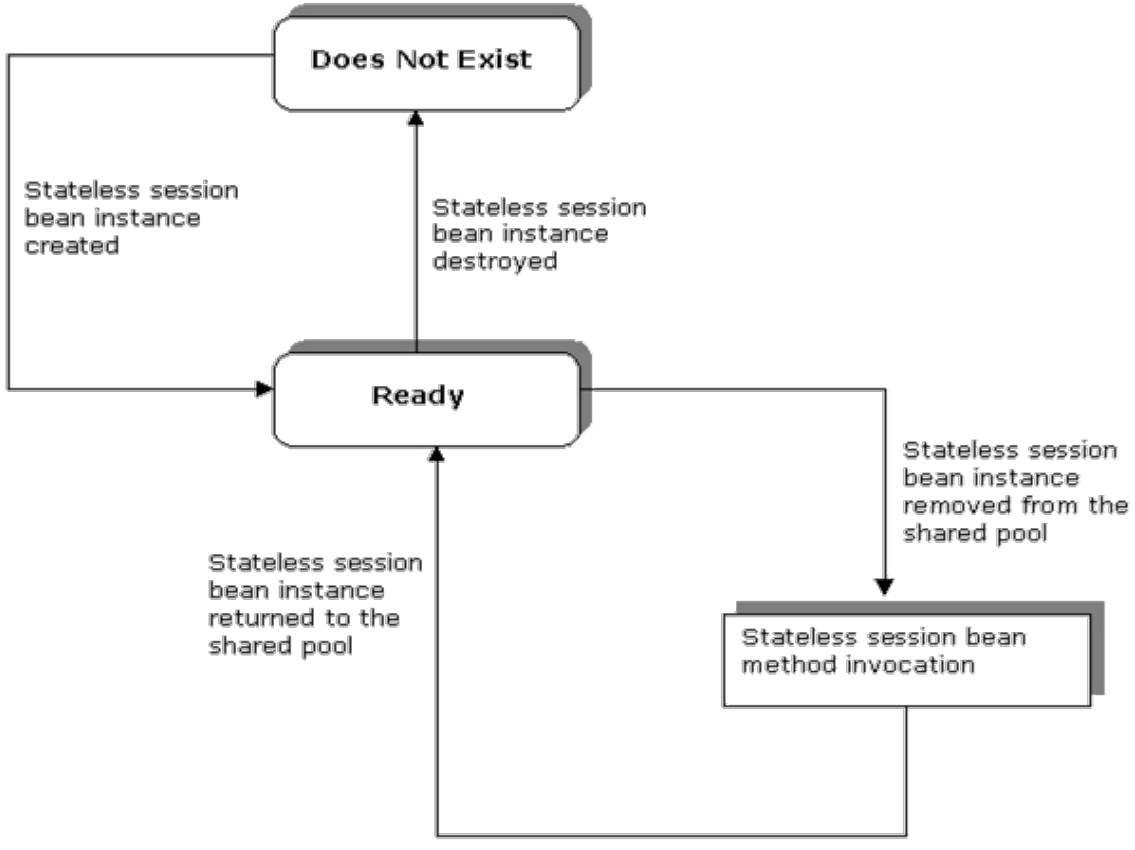
Las características de los Beans de Sesión con Estado son:

- Mantiene el estado del cliente durante la ejecución de diferentes operaciones de negocio en las aplicaciones empresariales.
- Reduce la escalabilidad de la aplicación ya que el contenedor EJB necesita gestionar un mayor numero de instancias del bean de sesión con estado para dar servicio a las solicitudes de multiples clientes.
- En las aplicaciones empresariales, el numero de instancias del bean de sesión con estado es igual al numero de clientes. Esto se debe a que cada instancia del bean de sesión con estado almacena el estado de un cliente en particular.

Beans de Sesión sin Estado

El ciclo de vida de un bean de sesión sin estado se compone de dos etapas:

- Etapa Listo
- Etapa No Existe



Beans de Sesión sin Estado

La Etapa Listo

- Al inicio de su ciclo de vida, un bean se sesión sin estado se encuentra en la etapa Listo.
- En esta etapa, una instancia de bean de sesión sin estado permanece en el grupo compartido, listo para dar servicio a las solicitudes del cliente. Un grupo compartido se refiere a un grupo de instancias de bean de sesión sin estado mantenido por el contenedor EJB para manejar las solicitudes del cliente.
- Un bean de sesión sin estado entra al grupo compartido al ser creado por el contenedor EJB.
- Si el número de instancias de bean de sesión sin estado es insuficiente para dar servicio a las solicitudes del cliente, el contenedor EJB crea nuevas instancias y las coloca en el grupo compartido.

Beans de Sesión sin Estado

Los pasos realizados por el contenedor EJB para crear una nueva instancia del bean de sesión sin estado, son:

1. Invoca al método **`newInstance`**`()`, el cual crea una nueva instancia de bean de sesión sin estado llamando al constructor predeterminado de bean de sesión sin estado.
2. Invoca al método **`setSessionContext`**`()` para asociar la instancia del bean con la información sobre el entorno en el cual se ejecutará la instancia del bean.
3. Invoca al método **`ejbCreate`**`()` definido en la clase del bean se sesión sin estado. El método **`ejbCreate`**`()` para un bean de sesión sin estado no contiene argumentos porque un bean de sesión sin estado no almacena la información de un cliente específico

Beans de Sesión sin Estado

La Etapa No Existe

- Al término de su ciclo de vida, un bean de sesión sin estado se encuentra en la etapa No Existe.
- En esta etapa, un bean de sesión sin estado es eliminado permanentemente del grupo compartido.

Beans de Sesión sin Estado

Contexto de Sesión EJB

- El contenedor EJB contiene información detallada de cada instancia del enterprise bean.
- La información incluye la referencia de la interfaz home de un bean de sesión sin estado, los permisos de seguridad del cliente y las transacciones actualmente asociadas con la instancia del bean. Toda esta información se almacena en el objeto de contexto EJB.
- Un enterprise bean utiliza un objeto de contexto EJB para acceder al estado del bean, como su información de transacción y seguridad.
- Hay un objeto de contexto EJB diferente correspondiente a cada tipo de enterprise bean.
- El objeto de contexto de sesión EJB habilita a los beans de sesión para interactuar con el contenedor EJB para realizar las siguientes funciones:
    - Recuperar la referencia a los objetos home
    - Recuperar los atributos de transacción
    - Establecer los atributos de transacción

Beans de Sesión sin Estado

- El paquete `javax.ejb` provee la interfaz `SessionContext` que le permite acceder a un objeto de contexto de sesión EJB.
- El método `setSessionContext()` devuelve la información sobre el entorno de un bean de sesión sin estado utilizando los siguientes métodos de la interfaz `SessionContext`:
  - `EJBObject getEJBObject()`: Devuelve la referencia al objeto EJB del bean de sesión en el cual se llama al método `getEJBObject()`.
  - `EJBLocalObject getEJBLocalObject()`: Devuelve la referencia al objeto EJB local del bean de sesión en el cual se llama al método `getEJBLocalObject()`.
  - `EJBHome getEJBHome()`: Devuelve la referencia al objeto home del bean de sesión en el cual se llama al método `getEJBHome()`.
  - `EJBLocalHome getEJBLocalHome()`: Devuelve la referencia al objeto home local del bean de sesión en el cual se llama al método `getEJBLocalHome()`.

Creando beans de Sesión sin Estado

Utilizando Archivos Java Para Crear un Bean de Sesión Sin Estado

- Usted necesita definir los tres siguientes archivos Java para crear un bean de sesión sin estado:
  - Interfaz home del bean de sesión sin estado: Contiene métodos para crear y eliminar las instancias del bean de sesión sin estado.
  - Interfaz remota del bean de sesión sin estado: Contiene los métodos de negocio implementados en la clase del bean de sesión sin estado.
  - Clase del bean de sesión sin estado: Implementa los métodos del ciclo de vida del bean de sesión sin estado y los métodos de negocio definidos en la interfaz remota del bean de sesión sin estado.

Creando beans de Sesión sin Estado

Creando la Interfaz Home de un Bean de Sesión Sin Estado

- La interfaz home declara los diferentes métodos del ciclo de vida de un bean de sesión sin estado, como **`remove`**`()` y **`create`**`()`.
- El contenedor EJB genera los objetos home extendiendo la interfaz home.
- Los clientes utilizan los objetos home del bean de sesión sin estado para crear y acceder a un objeto EJB.

Creando beans de Sesión sin Estado

- Usted puede crear dos tipos de interfaces home del bean de sesión sin estado:
  - Interfaz home remota
  - Interfaz home local
- Usted necesita extender la interfaz `EJBHome` paquete `javax.ejb`, para crear una interfaz home remota del bean de sesión sin estado.
- La interfaz `EJBHome` define los métodos siguientes invocados por los clientes para gestionar remotamente a un bean de sesión sin estado:
  - `EJBMetaData getEJBMetaData()`
  - `HomeHandle getHomeHandle()`
  - `void remove(Handle hd)`
- La interfaz de un bean de sesión sin estado declara al método `create()`.

Creando beans de Sesión sin Estado

- El método `create()` de la interfaz remota del bean de sesión sin estado inicia las excepciones `javax.ejb.CreateException` y `javax.ejb.RemoteException`.
- El tipo de devolución del método `create()` es del tipo interfaz remota del bean de sesión sin estado.
- La interfaz home remota de un bean de sesión debe satisfacer los siguientes requerimientos adicionales:
  - Necesita extender la interfaz `javax.ejb.EJBHome`.
  - Necesita declarar uno o más métodos `create()`. En el caso de los beans de sesión sin estado, solamente se requiere de un método `create()`.
  - Ésta necesita declarar los métodos `create()` correspondientes a los métodos `ejbCreate()` en la clase bean y el tipo de devolución de los métodos `create()` deben ser de tipo interfaz remota.
  - Esta necesita declara la cláusula `throws` del método `create()` como una `javax.ejb.CreateException` y `java.rmi.RemoteException`.

Creando beans de Sesión sin Estado

- Usted necesita extender la interfaz `EJBLocalHome` para crear una interfaz home local del bean de sesión sin estado.
- Usted necesita declarar el método `create()` en la interfaz home local de un bean de sesión sin estado.
- El método `create()` de la interfaz home local de un bean de sesión sin estado inicia las excepciones `javax.ejb.CreateException` y `javax.ejb.EJBException`.
- El tipo de devolución del método `create()` es de tipo interfaz local del bean de sesión sin estado.

Creando beans de Sesión sin Estado

La interfaz home local de un bean de sesión necesita cumplir con los siguientes requerimientos:

- No debe declarar métodos, los cuales inicien excepciones de tipo `RemoteException`.
- Puede tener super interfaces.
- Debe declarar uno o más métodos `create()` correspondientes a los métodos `ejbCreate()`en la clase del bean de sesión. El tipo de devolución de cada método `create()` debe ser de interfaz local.
- Para cada método `create()`, esta debe declarar las mismas excepciones declaradas en la cláusula inicia del método `ejbCreate()` correspondiente en la clase del bean.
- Debe declarar la `javax.ejb.CreateException` en la cláusula inicia de cada método `create()`.

Creando beans de Sesión sin Estado

Creando la Interfaz Remota de un Bean de Sesión Sin Estado

- La interfaz remota de un bean de sesión sin estado declara los métodos de negocio del bean que un cliente puede llamar. Estos métodos de negocio se implementan en la clase del bean de sesión sin estado.
- Usted necesita extender la interfaz `EJBObject` almacenada en el paquete `javax.ejb` para crear la interfaz remota del bean de sesión sin estado.
- Los métodos en la interfaz `EJBObject` se declaran como abstractos y su implementación puede ser provista en la interfaz remota.
- Los diferentes métodos declarados en la interfaz `EJBObject` son:
  - `EJBHome getEJBHome()`
  - `Handle getHandle()`
  - `boolean isIdentical(EJBObject obj)`
  - `void remove()`

Creando beans de Sesión sin Estado

- Usted también puede crear una interfaz local del bean de sesión sin estado que declare el método de negocio al que pueden llamar los clientes locales.
- Usted necesita declarar los métodos de negocio en la interfaz local de un bean de sesión sin estado de la misma manera en que se hace para la interfaz remota de un bean de sesión sin estado.
- La interfaz local de un bean de sesión sin estado debe cumplir los siguientes requerimientos:
  - Necesita extender la interfaz `javax.ejb.EJBLocalHome`.
  - No debe declarar los métodos de negocio con excepción, `java.rmi.RemoteException`, en su cláusula `throws`.
  - Debe declarar solamente los métodos de negocio, los cuales están implementados en el archivo de la clase del bean del bean de sesión sin estado.
  - Puede tener super interfaces que cumplan con los requerimientos de RMI/IIOP.

Creando beans de Sesión sin Estado

Creando la Clase Bean de Sesión Sin Estado

- La Clase Bean de Sesión Sin Estado:
  - Implementa los métodos del ciclo de vida utilizados por el contenedor EJB para controlar el ciclo de vida de un bean de sesión sin estado.
  - Implementa los métodos de negocio declarados en la interfaz remota del bean de sesión sin estado.
  - Se crea implementando la interfaz `SessionBean` del paquete `javax.ejb`.

Creando beans de Sesión sin Estado

- La interfaz `SessionBean` contiene los siguientes métodos para controlar el ciclo de vida de un bean de sesión sin estado:
  - `ejbActivate()`
  - `ejbPassivate()`
  - `ejbRemove()`
  - `setSessionContext(SessionContext ctx)`

Creando beans de Sesión sin Estado

- Una clase bean de sesión sin estado declara un constructor sin argumentos. El contenedor EJB invoca a este constructor para crear una instancia del bean de sesión sin estado.
- El contenedor EJB no activa o vuelve pasiva una instancia del bean de sesión sin estado, por lo tanto, usted debe proveer una implementación vacía para los métodos `ejbActivate()` y `ejbPassivate()` en la clase bean de sesión sin estado.
- La clase bean necesita cumplir con los siguientes requerimientos:
  - Debe implementar la interfaz `javax.ejb.SessionBean`.
  - Debe ser declarada como `public` y no como `final o abstract`.
  - Debe declarar un constructor `public` que no acepte argumentos.
  - No debe definir al método `finalize()`.

Creando beans de Sesión sin Estado

- Debe implementar los métodos de negocio que necesitan declararse como públicos y no como finales o estáticos. Estos métodos de negocio pueden iniciar excepciones de aplicación.
- Puede tener super clases y super interfaces.
- Debe implementar uno o más métodos `ejbCreate()` cumplan con los siguientes criterios:
  - Deben ser declarados como `public` y no como `final o static`.
  - Deben tener un tipo de devolución `void`.
  - Debe tener argumentos y valores de devolución compatibles con RMI/IIOP.

Creando beans de Sesión sin Estado

Compilando y Desplegando un Bean de Sesión Sin Estado

- Después de crear los archivos Java para un bean de sesión sin estado, usted necesita establecer la ubicación del archivo `j2ee.jar` existente en el directorio `Sun/Appserver/lib` en la ruta de clase del sistema. El comando utilizado para establecer la ubicación del archivo `j2ee.jar` es:
  - **`set classpath=.c:/Sun/Appserver/lib/j2ee.jar`**
- Después de establecer la ruta de clase, usted necesita compilar todos los archivos Java utilizando el compilador `javac`. El comando utilizado para compilar los archivos Java es:
  - **`javac <file_name>`**
- Los archivos de la clase Java compilados del bean de sesión sin estado se despliegan en el Servidor de la Aplicación J2EE1.4 usando la utilidad de la herramienta **`deploytool`**.

Creando beans de Sesión sin Estado

- El Asistente Enterprise Bean de la utilidad de la herramienta deploytool se utiliza para desplegar un bean de sesión sin estado. La utilidad de la herramienta deploytool empaqueta los archivos compilados de la clase Java en un archivo JAR.
- La utilidad de la herramienta deploytool genera automáticamente el código del descriptor de despliegue de un bean de sesión sin estado.

Creando beans de Sesión sin Estado

Accediendo a un Bean de Sesión Sin Estado

- Un cliente accede a los métodos de negocio de un bean de sesión sin estado utilizando las referencias de sus interfaces home y remota.
- Un bean de sesión sin estado puede ser accedido tanto por los clientes de Red como de la Aplicación. Los clientes de la Red consisten en Páginas de Servidor Java (JSP) y servlets mientras que los clientes de Aplicación consisten en clases Java autónomas.
- Los pasos realizados por un cliente para acceder a un bean de sesión sin estado son:
  1. Ubica el bean de sesión sin estado en el Servidor de la Aplicación J2EE 1.4.
  2. Recupera las referencias de las interfaces home y remota de un bean de sesión sin estado.

Creando beans de Sesión sin Estado

Ubicando a un Bean de Sesión Sin Estado

- Un cliente ubica a un bean de sesión sin estado en el Servidor de la Aplicación J2EE 1.4 utilizando la Interfaz del Directorio de Nombres de Java (JNDI).
- Para ubicar a un bean de sesión sin estado, el cliente realiza los siguientes pasos:
  - Crea un contexto inicial de nombres utilizando la interfaz `InitialContext` de JNDI.
  - Ubica al objeto home del bean de sesión sin estado desplegado utilizando el método `lookup()`. Este método devuelve la referencia del objeto home, el cual es una implementación de la interfaz home del bean de sesión sin estado.

Creando beans de Sesión sin Estado

Recuperando las Referencias de las Interfaces de un Bean de Sesión Sin Estado

- Después de ubicar al objeto home del bean de sesión sin estado, un cliente recupera la referencia al objeto EJB creado por el contenedor EJB.
- Los clientes no tienen acceso directo a una clase bean de sesión sin estado. Estos pueden invocar a los métodos de negocio de un bean utilizando la referencia del objeto EJB el cual es una implementación de la interfaz remota del bean de sesión sin estado.
- Los clientes utilizan el método `narrow()` de la interfaz `PortableRemoteObject` para recuperar la referencia del objeto EJB.
- Los clientes locales recuperan la referencia de la interfaz home local del bean de sesión sin estado utilizando el método `lookup()` de la interfaz `InitialContext`

Creando beans de Sesión sin Estado

Un cliente llama al método **`create`**( ) en la interfaz home del bean de sesión sin estado para recuperar la referencia de la interfaz remota del bean de sesión sin estado.

Resumen

En esta lección, usted aprendió:

- Los beans de sesión son enterprise beans desplegados en el contenedor EJB que realizan funciones de negocio, a nombre de los clientes.
- En las aplicaciones empresariales distribuidas, los clientes utilizan los métodos definidos en los beans de sesión para acceder a los servicios remotos definidos en el servidor.
- Existen dos tipos de beans de sesión, beans de sesión con estado y beans de sesión sin estado.
- Un bean de sesión sin estado no mantiene el estado del cliente a través de las llamadas de método. Cuando se completa la ejecución de un método del bean, se deja de retener la información del cliente.
- Un bean de sesión con estado mantiene el estado del cliente durante la ejecución de las operaciones de negocio.
- El ciclo de vida de un bean de sesión sin estado consiste de dos etapas, Listo y No Existe.

Resumen

- Para crear un bean de sesión sin estado, usted necesita realizar las siguientes tareas:
  - Crear una interfaz home remota o la interfaz home local del bean de sesión sin estado
  - Crear una interfaz remota o local del bean de sesión sin estado
  - Crear la clase bean de sesión sin estado
- Un cliente del bean de sesión sin estado puede acceder al bean ubicando primero al bean de sesión sin estado en el servidor J2EE 1.4 y luego recuperando las interfaces home y remota del bean de sesión sin estado. El cliente llama entonces a los métodos de negocio de la interfaz remota para realizar las operaciones de negocio.


" GRACIAS "