Especialista en Aplicaciones y Servicios Web con Java Enterprise

Enterprise JavaBeans

Sesión 1:

Introducción a los Enterprise Beans

Índice

- Arquitectura Enterprise JavaBeans
- Servicios del servidor de aplicaciones
- Funcionamiento de los componentes
- Tipos de beans
- Un ejemplo (práctico) de bean de sesión sin estado

Arquitectura EJB

- La arquitectura EJB se propone como un marco de computación basada en componentes distribuidos
- Historia:
  - EJB 1.0 (Abril 1998)
  - EJB 1.1 (Diciembre 1999) - Descriptores XML
  - EJB 2.0 (Agosto 2001) - Primer estándar (JSR 19)
  - EJB 2.1 (Noviembre 2003) - Interfaces locales y servicios web
  - EJB 3.0 (Mayo 2006) - POJOs y anotaciones

¿Qué es un Enterprise JavaBean?

- Desde el punto de vista de un cliente, un EJB es:
  - Un objeto que reside en un contenedor EJB remoto.
  - Proporciona acceso a un conjunto de servicios definidos por su interfaz de negocio.
  - El contenedor EJB recubre la interfaz de negocio con un conjunto de servicios añadidos (seguridad, transaccionalidad, concurrencia, escalabilidad) que se implementan de forma transparente al desarrollador del EJB.

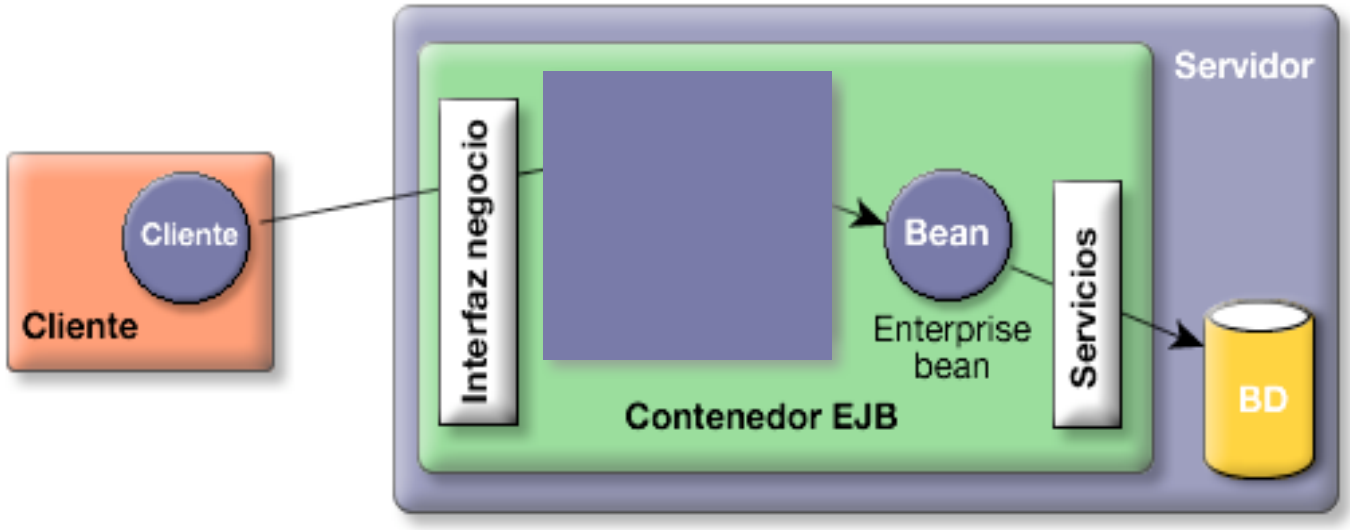
Desarrollo basado en componentes

- Los componentes (*enterprise beans*) permiten reusar código y datos.
- Los componentes se despliegan en un servidor (contenedor EJB).
- Componente = servicios del objeto + servicios del contenedor EJB
- Ventajas del desarrollo basado en componentes:
  - Reusabilidad
  - Modularidad
  - Interoperabilidad entre aplicaciones distribuidas

Características de los componentes

- Permiten el desarrollo de aplicaciones débilmente acopladas.
- Su comportamiento está especificado por interfaces.
- Su funcionamiento se puede configurar en tiempo de despliegue de forma declarativa (descriptor de despliegue).
- Los componentes EJB son multiplataforma (WODE: Write Once Deploy Everywhere).

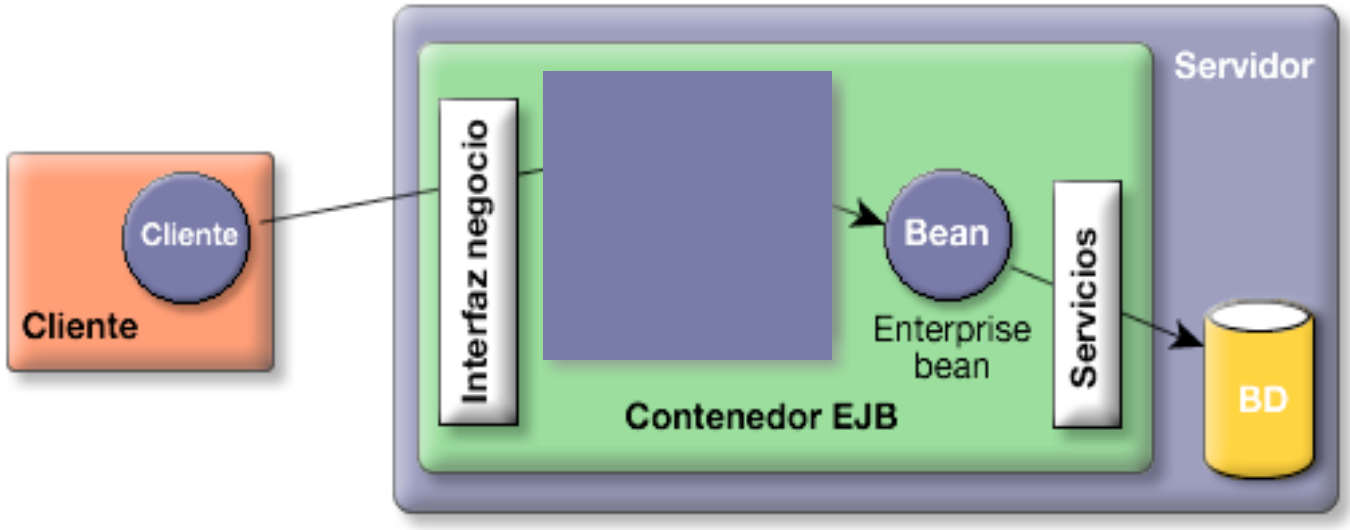
Funcionamiento de un bean (1)



Objeto “cortafuegos” - EJB Object

- Carácterísticas del contenedor EJB:
  - El cliente de un componente nunca hablará directamente con la implementación del componente (bean).
  - Un objeto remoto (EJB Object) hará de "cortafuegos" y permitirá que el contenedor interponga sus servicios en cada llamada.
  - El EJB Object ofrece los mismos métodos de negocio que el bean.

Funcionamiento de un bean (2)



Servicios del contenedor EJB

- Transacciones
- Seguridad
- Concurrencia
- Threading
- Gestión y pooling de recursos
- Persistencia
- Gestión de mensajes
- Escalabilidad

Interfaz Home

- Un concepto importante en la arquitectura EJB es el concepto de interfaz Home.
- Una clase Home es similar a una factoría con métodos para crear beans remotos.
- Hay una clase Home por cada tipo de EJB que se define.
- Ejemplo: **`SaludoHome.java`**

SaludoHome.java

```
package es.ua.jtech.ejb.beans;


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


public interface SaludoHome extends EJBHome {
         public Saludo create() throws RemoteException, CreateException;
}
```

Interfaz componente

- La interfaz componente proporciona al cliente remoto los métodos que implementa el componente enterprise.
- Ejemplo: **`Saludo.java`**

Saludo.java

```
package es.ua.jtech.ejb.beans;

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

public interface Saludo extends EJBObject {
   public String saluda() throws RemoteException;
   public SaludoTO getSaludo(int numDatos) throws RemoteException;
}
```

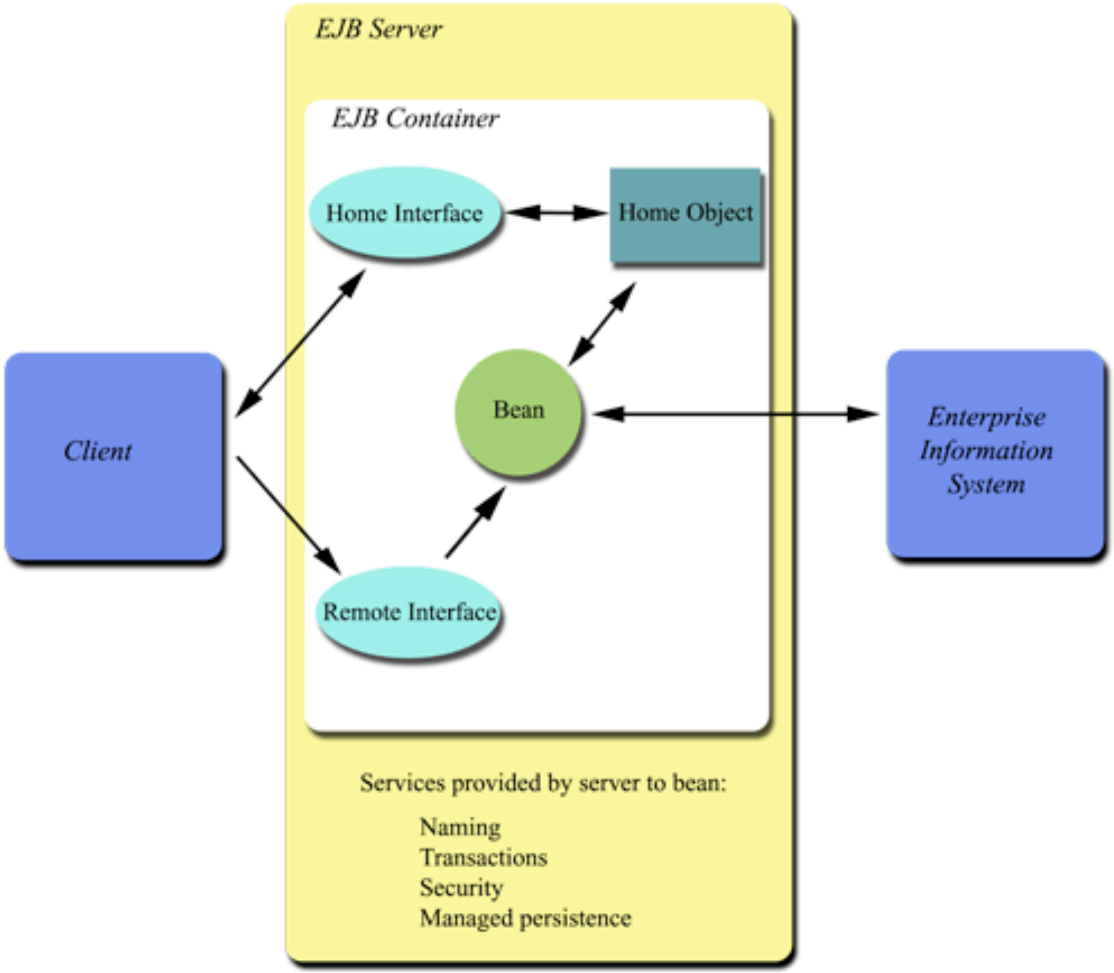
Un ejemplo de cliente (1)

```
public class SaludoClient {
   public static void main(String[] args) {
      try {
         System.out.println("Fecha actual: " + (new Date()).toString());
         Context jndiContext = getInitialContext();
         Object obj = jndiContext.lookup("SaludoBean");
         SaludoHome home = (SaludoHome) narrow(obj, SaludoHome.class);
         obj = home.create();
         Saludo saludo = (Saludo) narrow(obj, Saludo.class);

         System.out.println("El bean saludo dice: " + saludo.saluda());
         System.out.println("Voy a obtener 1 dato...");
         SaludoTO saludoTO = saludo.getSaludo(1);
         System.out.println("Mensaje: " + saludoTO.getMensaje());
         System.out.println("Fecha creacion enterprise bean: "
               + saludoTO.getFecha().toString());
         List datos = saludoTO.getDatos();
         System.out.println("Los datos tienen " + datos.size() + " elementos");
         System.out.println("Fecha actual: " + (new Date()).toString());

      } catch (Exception e) {
         e.printStackTrace(); // [7]
      }
   }
   ...
```

Acceso al contexto JNDI y narrow

```
private static Context getInitialContext() // [8]
        throws javax.naming.NamingException, IOException {
    Properties p = new Properties();
    p.put(Context.INITIAL_CONTEXT_FACTORY,
          "weblogic.jndi.WLInitialContextFactory");
    p.put(Context.PROVIDER_URL, "t3://localhost:7001");
    return new InitialContext(p);
  }

  private static Object narrow(Object obj, Class clase) { //[9]
    return PortableRemoteObject.narrow(obj, clase);
  }
```

Diagrama completo de la arquitectura



EJB vs. RMI

- RMI es la base del funcionamiento de los componentes EJB
- Con RMI es posible acceder a objetos remotos
- Con EJB es posible acceder a objetos remotos gestionados por un contenedor que proporciona servicios

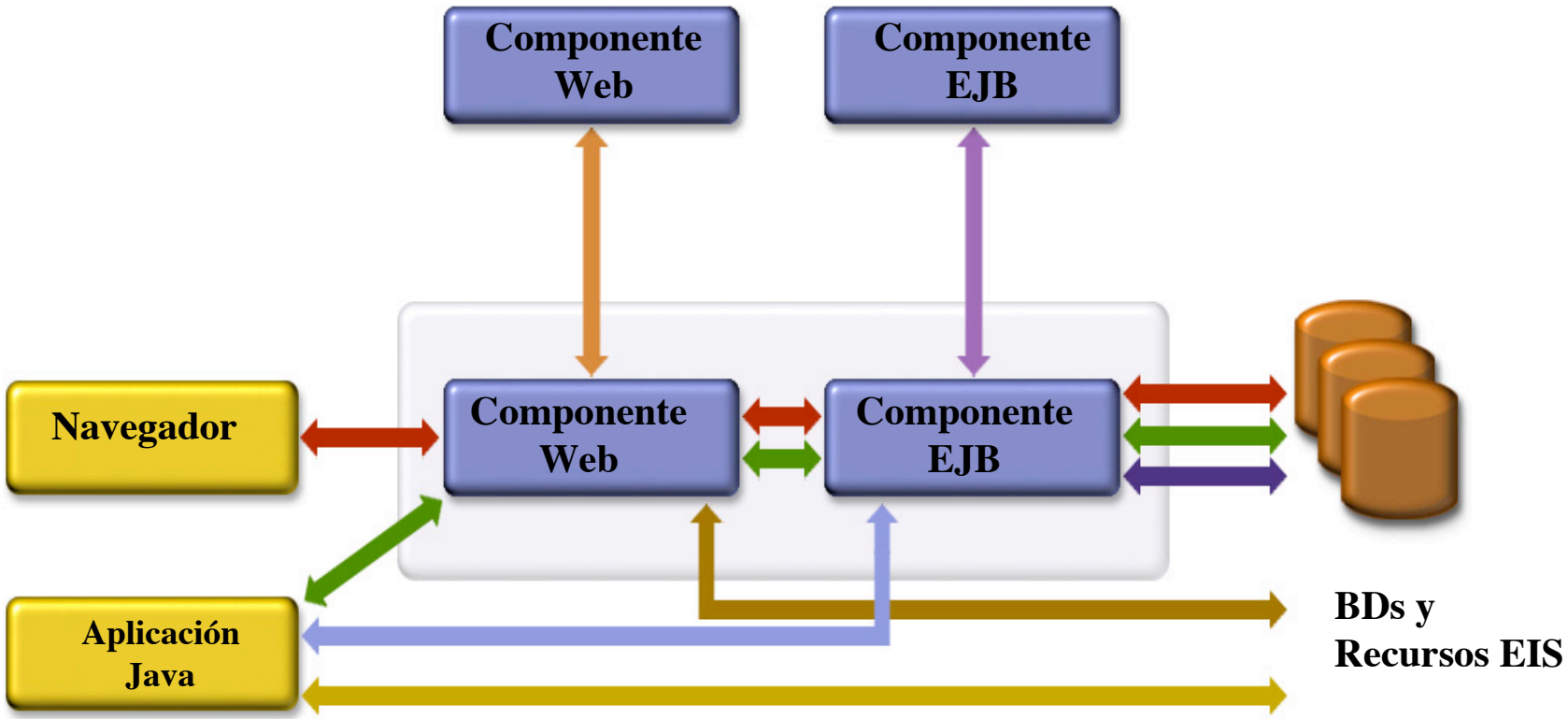
EJB vs. Servicios Web

- Ambos proporcionan servicios remotos
- Servicios Web:
  - Se basan en peticiones HTTP
  - Los parámetros y las respuestas se codifican en ficheros XML (texto) - complicado manejo del cliente
  - Los servicios (seguridad,...) son bastante escasos
  - Facilidad de conexión (conexión HTTP)
- EJB:
  - Se basa en RMI
  - Los parámetros y las respuestas son objetos Java - facilidad de programar
  - Amplios servicios proporcionados por el servidor de aplicaciones
  - Conexión mediante JNDI utilizando puertos específicos

Tipos de componentes

- Beans de sesión (*Session Beans)*
  - Proporcionan servicios (capa de negocio)
  - *Beans de sesión sin estado*
  - *Beans de sesión con estado*
- Beans de entidad (*Entity Beans)*
  - Proporcionan acceso a datos y relaciones entre ellos (capa de persistencia)
  - *Con persistencia gestionada por el bean (BMP)*
  - *Con persistencia gestionada por el contenedor (CMP)*
- *Beans dirigidos por mensajes (Message Driven Beans)*

EJB en J2EE


Componente Web
Componente EJB
Navegador
Componente Web
Componente EJB
Aplicación Java
BDs y Recursos EIS

¿Preguntas?