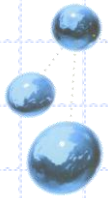


CC68J APLICACIONES EMPRESARIALES CON JEE

ENTERPRISE JAVA BEANS

1. Introducción al estándar

Profesores:

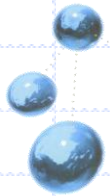
- Andrés Farías

fcfm

v1.0

Objetivos de esta sección

1. Entender los conceptos generales de EJB 3.0.
2. Entender el soporte para persistencia del modelo EJB.
3. Entender de qué se trata la mensajería asíncrona.
4. Entender el fin de los Servicios Web (Web Services) en el marco de EJB's.
5. Conocer un poco del ejemplo conductor de usado para EJB's.

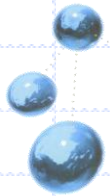


Introducción

Enterprise JavaBeans

Conceptos Generales

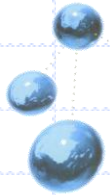
- Tecnología por excelencia para el desarrollo de componentes en la parte servidora con la plataforma **JEE**.
- Deben de conformar a la especificación de **JEE** y sólo se pueden ejecutar en un contenedor de EJBs.
  - ✓ Última especificación **EJB 3.0** disponible en: http://java.sun.com/products/ejb/docs.html
- Los contenedores de **EJBs** son lo que hacen a esta tecnología tan atractiva, ya que ofrecen soporte para:
  - ✓ Transacciones
  - ✓ Seguridad
  - ✓ Persistencia
- … tanto de manera programática como declarativa



Enterprise JavaBeans

Definición

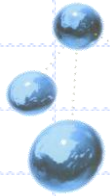
- Definición oficial de Sun:
  - ✓ La arquitectura **Enterprise JavaBeans** es una arquitectura de componentes para el desarrollo y despliegue de aplicaciones de negocio distribuidas basadas en componentes. Aplicaciones escritas utilizando la arquitectura Enterprise JavaBeans son **escalables**, **transaccionales**, y **seguras** para multi-usuarios. Estas aplicaciones podrían ser escritas una vez, y luego desplegadas en cualquier servidor de plataforma que soporte la especificación Enterprise JavaBeans.
- En resumen:
  - ✓ **Enterprise JavaBeans** es un estándar de un modelo de componentes en el servidor para aplicaciones de negocio distribuidas.



Enterprise JavaBeans

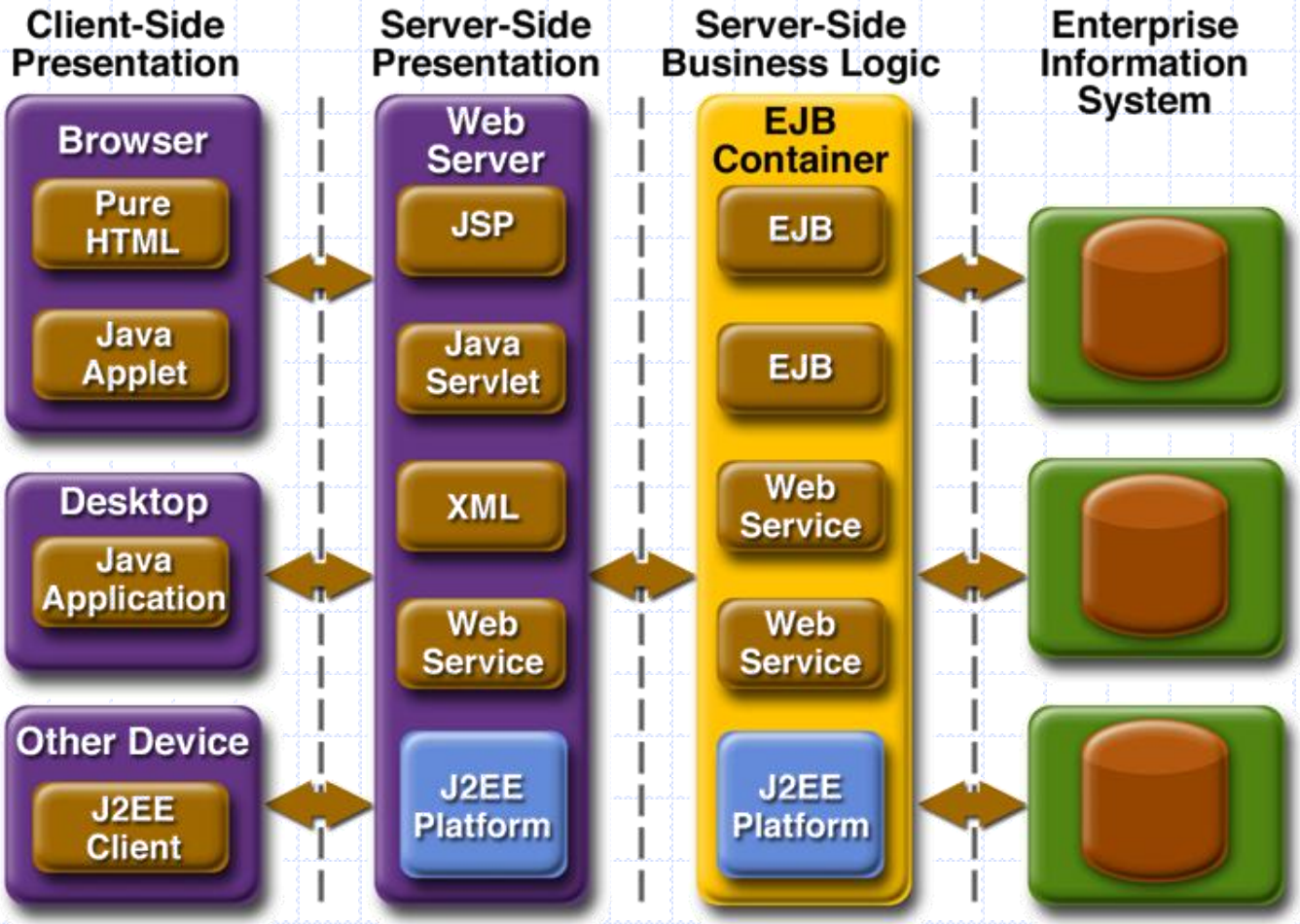
Beneficios

- Aplicaciones basadas en EJBs nos permiten concentrarnos en la lógica de negocio, sin preocuparnos de transacciones y connection pooling que son provistos por el contenedor.
- Los EJBs son *componentes* lo que juega a favor de la reutilización de manera importante.
- Clara separación entre desarrollo, explotación y administración de una aplicación EJB.
- El contenedor de EJBs gestiona transacciones, detalles de manejo de estado, multi-threading, connection pooling, seguridad y otros detalles de bajo nivel que el desarrollador no necesita conocer.



Enterprise JavaBeans

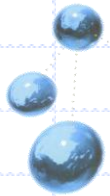
Ubicación en la Arquitectura



Enterprise JavaBeans

Tecnologías usadas

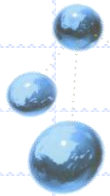
- EJBs hacen uso:
  - ✓ RMI
  - ✓ RMI-IIOP
  - ✓ JNDI (Java Naming and Directory Interface)
  - ✓ JMS (Java Messaging Service)



Contenedores de EJBs

Implementaciones conocidas

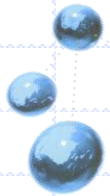
- JBoss:
  - ✓ Contenedor EJB certificado.
- Oracle WebLogic:
  - ✓ Contenedor EJB certificado.
- IBM Websphere:
  - ✓ Contenedor EJB certificado.
- Oracle10g:



Enterprise Java Beans

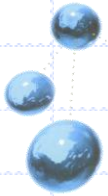
Categorías de EJBs

- Entity Beans:
  - ✓ Representan datos de negocio y proveen acceso a datos a través de métodos.
- Session Beans:
  - ✓ Modelan procesos de negocio que son accedidos de manera síncrona.
- Message-driven Beans:
  - ✓ Modelan procesos de negocio que son accedidos de manera asíncrona, permiten el uso de JMS desde EJBs.



Interactuando con sistemas de almacenamiento

Persistencia & Entity Beans



Persistencia y Entity Beans

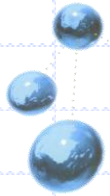
Persistencia en EJB 3.0

- En **EJB 3.0**, la persistencia es una abstracción de un nivel de abstracción superior a **JDBC**.
- La capa de persistencia permite mapear objetos a la BDD, para consultar, cargar, actualizar o eliminar.
- En versiones anteriores la capa de persistencia era parte de la plataforma EJB. A partir de la versión 3.0 la persistencia tiene su propia especificación llamada **Java Persistance API**, o **JPA**.

Persistencia y Entity Beans

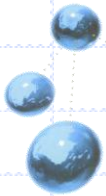
JPA & Entity Beans

- La API de persistencia es responsable de definir una manera de mapear objetos Java regulares, planos y viejos (**POJO** para *Plain Old Java Objects*), a la base de datos. Estos POJO's son llamados **Entity Beans** o **Beans de Entidad**.
  - ✓ **Entity Beans** son mapeados, usando metadata de Java Persistance, a la base de datos.
  - ✓ Pueden ser insertados, actualizados o cargados en la BDD sin programar código JDBC.
  - ✓ JPA define un lenguaje de consulta que paraleliza las ventajas de SQL, pero de un punto de vista orientado a objetos, más que relacional.



Intercambio de mensajes distribuido y asíncrono

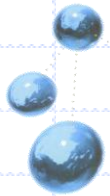
Mensajería Asíncrona



Mensajería asíncrona

Mensajes

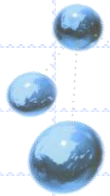
- Un sistema de mensajería asíncrona permite a dos o más aplicaciones intercambiar información en la forma de *mensajes*.
- Un mensaje es un paquete auto-contenido de datos de negocio y encabezados de ruteo de redes.
  - ✓ Los datos de negocio pueden contener cualquier tipo de información relativa a una transacción de negocio.
  - ✓ En sistemas empresariales, los mensajes cumplen el rol de informarle a una aplicación de un evento u ocurrencia en otro sistema.



Mensajería asíncrona

MOM: Message-oriented Middleware

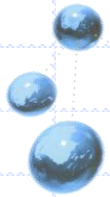
- Mensajes asíncronos podrían ser transmitidos desde una aplicación a otra de la red usando **Message-oriented Middleware**, **MOM**.
- Productos MOM aseguran que los mensajes se distribuyen apropiadamente entre aplicaciones. Más aun, MOM usualmente provee:
  - ✓ Tolerancia a fallas
  - ✓ Balanceo de carga,
  - ✓ Escalabilidad, y
  - ✓ Soporte para empresas que necesitan intercambiar de manera segura una cantidad importante de mensajes.



Mensajería asíncrona

JMS & Message Driven Beans

- Las aplicaciones intercambian mensajes a través de canales virtuales llamados *destinos* (*destinations*). Los mensajes están dirigidos a *destinos*, no a una aplicación específica.
  - ✓ Se utiliza entonces un patrón arquitectural *publicador-subscriptor*.
  - ✓ Remitente y receptores de mensajes no están relacionados el uno con el otro de ninguna manera.
- EJB integra MOM en su modelo de componentes a través de **Java Message Service** (**JMS**) y un nuevo componente llamado **Message-driven Bean**.



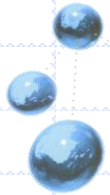
Exponiendo métodos de EJB's.

Web Services

Web Services

JMS & Message Driven Beans

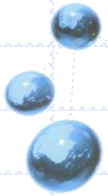
- Algunas definiciones de **Web Services** (**WS**):
  - ✓ "A substrate for building distributed applications using software running on different operating systems and devices" (Tim Ewald, "The Web Services Idea," July 12, 2002, Microsoft.com)
  - ✓ "Self-contained, self-describing, modular applications that can be published, located, and invoked across the Web." (Doug Tidwell, "Web services: the Web's next revolution," November 29, 2000, IBM.com)
- Nos quedaremos con la siguiente definición:
  - ✓ Web services son aplicaciones distribuidas que usan SOAP y WSDL para intercambiar información en forma de documentos XML.
- Donde:
  - ✓ **SOAP** 1.1 (**Simple Object Access Protocol**) es un gramática XML que da soporte a un protocolo de aplicaciones utilizado en RPC y mensajería asíncrona.
  - ✓ **WSDL** 1.1 (**Web Service Description Language**) es una interfaz basada en XML Inteface Definition Language (IDL) que puede ser usada para describir servicios web, incluyendo el tipo de formato del mensaje esperado, el protocolo de comunicación usado, y la dirección internet del servicio web.



Web Services

JMS & Message Driven Beans

- EJB 3.0 permite que JavaBeans puedan ser expuestos como servicios web para que sus métodos puedan ser invocados por otras aplicaciones J2EE así como aplicaciones escritas en otros lenguajes.
- Los Web Services en EJB 3.0 soporta tanto mensajería de estilo RPC y estilo Documentos.
- El soporte para Web Services está basado en una API de Web Services: JAX-WS.



Cruseros TITAN

El ejemplo conductor

Ejemplo-hilo conductor

Cruceros Titan

- Los ejemplos se apoyarán en un negocio imaginario, una línea de cruceros llamada *Cruceros Titan*.
  - ✓ Tiene cabinas de barco similares a cuartos de hotel.
  - ✓ Sirve comidas como un restaurant.
  - ✓ Ofrece diversas oportunidades de recreación
  - ✓ Necesita interactuar con otras entidades de viaje.
- Este tipo de negocio es un buen candidato para un sistema de objetos distribuídos, ya que los usuarios del sistema están geográficamente dispersos.
  - ✓ Agencias de negocio que reservan pasajes en naves Titan necesitan acceder al sistema de reservación.
  - ✓ Soportar eventualmente cientos de agencias de viaje requiere un sistema transaccional robusto para asegurar a los agentes que el acceso y las reservas son seguras.

Preguntas?