Enterprise JavaBeans es un modelo de componentes de servidor estándar para los monitores de transacciones de componentes

Un [monitor](http://www.monografias.com/trabajos5/losperif/losperif2.shtml#moni) de transacciones es un conjunto de uno o más componentes (no necesariamente en el sentido de componentes de objetos) que brindan el soporte para el diseño, desarrollo, configuración y operación de confiables aplicaciones de transacciones distribuidas. Esto significa que de alguna manera se debe garantizar las propiedades ACID para las aplicaciones, y también incluye la puesta en marcha de los procesos [servidores](http://www.monografias.com/trabajos12/rete/rete.shtml), la canalización de los mensajes de solicitud/respuesta y algún tipo de [supervisión](http://www.monografias.com/trabajos13/conce/conce.shtml) y [equilibrio](http://www.monografias.com/trabajos/tomadecisiones/tomadecisiones.shtml) de cargas..

Para llevar a cabo de mejor forma su [función](http://www.monografias.com/trabajos7/mafu/mafu.shtml) es deseable que una transacción tenga las siguientes características:

1. Atomicidad. Una transacción debe ser atómica. A pesar de que una transacción está compuesta por un número cualquiera de eventos, el sistema las debe considerar como una única operación, la cual puede tener éxito; en tal caso se hacen permanentes los cambios generados por cada evento componente de la transacción; o fracaso, en este caso el sistema queda en el mismo [estado](http://www.monografias.com/trabajos12/elorigest/elorigest.shtml), como si la transacción nunca hubiera ocurrido.

2. Consistencia. Todos los cambios provocados por la transacción deben dejar al sistema en un estado correcto. El sistema es llevado desde un estado válido a otro estado válido, [producto](http://www.monografias.com/trabajos12/elproduc/elproduc.shtml) de la [acción](http://www.monografias.com/trabajos35/categoria-accion/categoria-accion.shtml) de una transacción.

3. Aislamiento. Las transacciones que se ejecutan concurrentemente no se ven afectadas unas con otras. Si una transacción A cambia un sistema de un estado E1 a un estado

E2, una transacción B siempre verá al sistema en un estado E1 o E2, pero nunca en un estado intermedio.

4. Durabilidad. Si una transacción es terminada en forma exitosa los efectos serán permanentes.

Las aplicaciones escritas utilizando la arquitectura Enterprise JavaBeans son escalables, transaccionales y multiusuario seguro. Estas aplicaciones se pueden escribir una vez y luego desplegar en cualquier plataforma de servidor que soporta la especificación de JavaBeans Enterprise

El término "modelo de componentes" tiene muchas interpretaciones diferentes. Enterprise JavaBeans especifica un modelo de componentes del lado del servidor. El uso de un conjunto de clases e interfaces de los paquetes javax.ejb, los desarrolladores pueden crear, ensamblar y desplegar los componentes que se ajustan a la especificación EJB.

El original JavaBeans ™, el paquete java.beans en el API de Java central, es también un modelo de componentes, pero no es un modelo de componentes del lado del servidor como EJB. De hecho, aparte de compartir el nombre "JavaBeans", estos dos modelos de componentes no tienen ninguna relación. Gran parte de la literatura se ha referido a EJB como una extensión de los JavaBeans original, pero esto es una falsedad. Aparte de en el nombre compartido, y el hecho de que ambos son modelos de componentes de Java, los dos API sirven muy diferentes propósitos. EJB no se extiende o utilizar el modelo original componente JavaBeans.

Dado que los JavaBeans y Enterprise JavaBeans son completamente diferentes, ¿por qué son los dos modelos llamados componentes? En este contexto, un modelo de componente define un conjunto de interfaces y clases en la forma de paquetes de Java que deben ser utilizados de una manera particular para aislar y encapsular un conjunto de funcionalidad. Una vez que se define un componente, se convierte en una pieza independiente de software que puede ser distribuido y utilizado en otras aplicaciones. Un componente es desarrollado para un propósito específico, pero no una aplicación específica. En los JavaBeans original, un componente podría ser un botón pulsador o de hoja de cálculo que puede ser utilizado en cualquier aplicación de interfaz gráfica de usuario de acuerdo con las reglas especificadas en el modelo original componente JavaBeans. En EJB, un componente podría ser un objeto de negocio del cliente que puede ser desplegado en cualquier servidor EJB y se utiliza para desarrollar cualquier aplicación de negocios que necesita un objeto de negocio del cliente.

tecnologías de objetos distribuidos

EJB es un modelo de componente para los monitores de transacción de componentes, que se basan en tecnologías de objetos distribuidos. Sistemas de objetos distribuidos son la base para las arquitecturas de tres niveles modernos. En una arquitectura de tres capa, la lógica de la presentación reside en el cliente (primer nivel), la lógica de negocio en el nivel intermedio (segundo nivel), y otros recursos, tales como la base de datos, residen en el backend (tercer nivel).

