

## CURSO JAVA EE

# MANEJO DE TRANSACCIONES



Ing. Ubaldo Acosta

Por el experto: Ing. Ubaldo Acosta



## CURSO JAVA EE

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

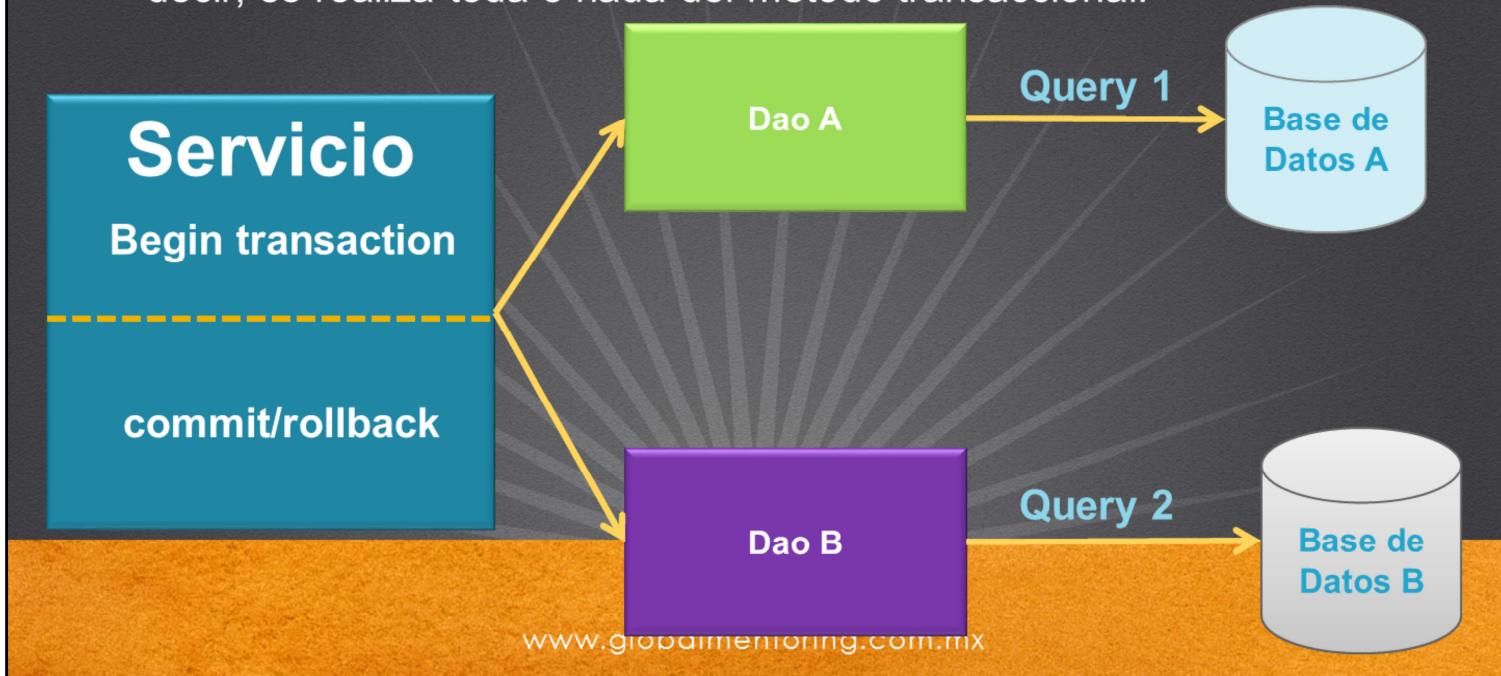
Hola, te saluda nuevamente Ubaldo Acosta. Espero que estés listo para comenzar con esta lección..

Vamos a estudiar el tema de Manejo de Transacciones en Java EE.

¿Estás listo? ¡Vamos!

## ¿QUÉ ES UNA TRANSACCIÓN?

- Una transacción se conoce como una unidad de trabajo atómica, es decir, se realiza toda o nada del método transaccional.



El Manejo Transaccional es uno de los temas cruciales en cuanto a requerimientos para aplicaciones empresariales. Esta motivación surge debido a que en todo sistema empresarial nos interesa mantener la integridad de nuestra información, con esto en mente es que surge el tema de transacciones.

En la figura podemos observar un método de servicio que ejecuta llamadas a más de un DAO, y a su vez cada DAO modifica el estado de la base de datos al escribir y/o modificar su información.

El objetivo de una transacción es ejecutar todas las líneas de código de nuestro método y guardar finalmente la información en un repositorio, por ejemplo en nuestro caso, una base de datos. Esto se conoce como **commit** de nuestra transacción.

Si por alguna razón algo fallara en nuestro método de Servicio, se daría marcha atrás a los cambios realizados en la base de datos. Esto se conoce como **rollback**.

Lo anterior permite que nuestra información, ya sea que se una única base de datos o no, esté íntegra, y no exista posibilidad de datos corruptos por errores o fallas en la ejecución de nuestros métodos Java.

Por ejemplo, imaginemos un sistema de venta de boletos de avión por internet. En este caso, los pasos necesarios para reservar un boleto son:

- 1) Verificar boletos disponibles
- 2) Reservar un boleto
- 3) Realizar el pago
- 4) Recibir los datos del boleto

Si por alguna razón alguno de los pasos falla, entonces el boleto no se debe considerar como un boleto vendido, sino como un boleto disponible. En caso de que todo se haya realizado exitosamente, entonces el boleto ya no está disponible y se actualiza el estado de la base de datos para reflejar un boleto menos para los nuevos usuarios interesados en comprar un boleto de avión.

Así que los métodos que hagamos partícipes de una transacción se ejecutarán como una sola acción, lo que garantiza que se ejecutan por completo, o si fallan, no se persista ninguna de la información. Los EJB al ejecutarse en un contenedor Java empresarial, por default son transaccionales, por lo tanto, cualquier método de un EJB maneja este concepto en automático.

# CARACTERÍSTICAS DE UNA TRANSACCIÓN

Las características de una transacción tienen el acrónimo ACID:

**Atomic:** Las actividades de un método se consideran como una unidad de trabajo. Esto se conoce como **Atomicidad**.

**Consistent:** Una vez que termina una transacción la información queda en estado consistente, ya que se realiza todo o nada.

**Isolated:** Múltiples usuarios pueden utilizar los métodos transaccionales, sin embargo debemos prevenir errores por accesos múltiples, aislando en la medida de lo posible nuestros métodos transaccionales.

**Durable:** Sin importar si hay una caída del servidor, una transacción exitosa debe guardarse y perdurar posterior al término de una transacción.

## CURSO JAVA EE

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

Las características de una transacción tienen el acrónimo ACID:

**Atomicidad:** Las actividades de un método se consideran como una unidad de trabajo. Esto se conoce como **Atomicidad**. Este concepto asegura que todas las operaciones en una transacción se ejecuta todo o nada.

Si todas las instrucciones o líneas de código de un método transaccional son ejecutadas con éxito, entonces al finalizar se realiza un **commit**, es decir, guardado de la información.

Si alguna de las instrucciones falla se realiza un **rollback**, es decir, ninguna de la información es guardada en la base de datos o el repositorio donde se persiste dicha información.

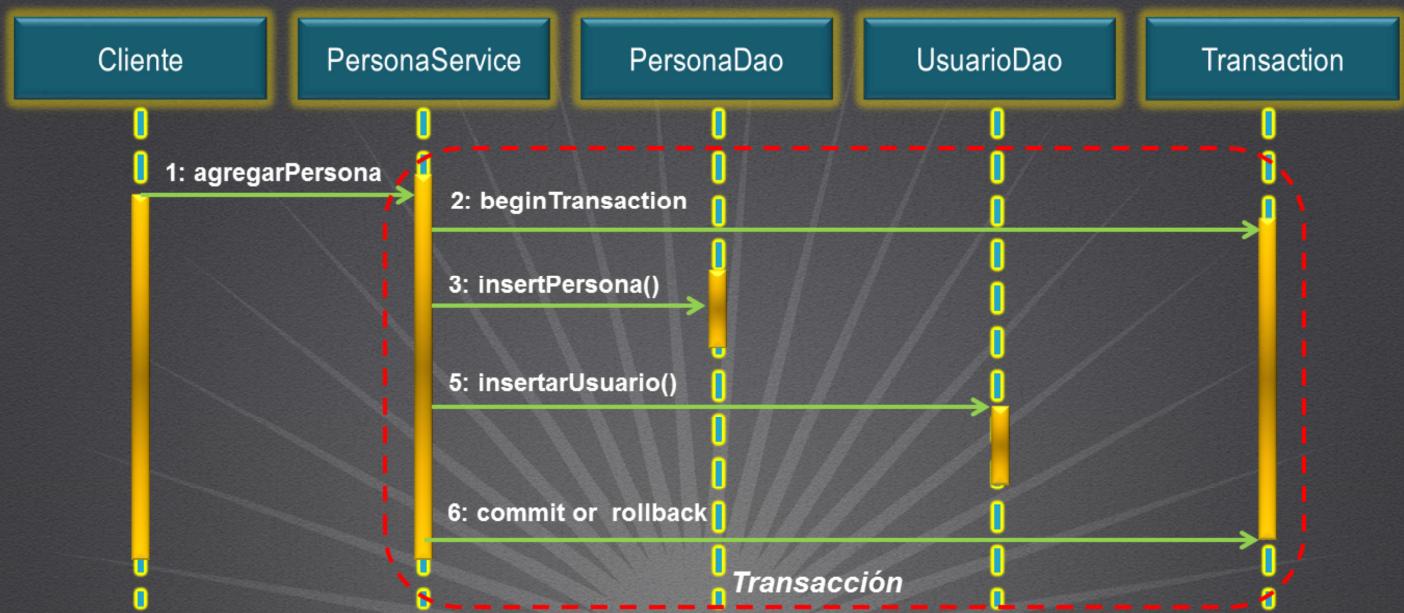
**Consistente:** Una vez que termina una transacción (sin importar si ha sido exitosa o no) la información queda en estado **consistente**, ya que se realizó todo o nada, y por lo tanto los datos no deben estar corruptos en ningún aspecto.

**Aislado:** Múltiples usuarios pueden utilizar los métodos transaccionales, sin afectar el acceso de otros usuarios. Sin embargo debemos prevenir errores por accesos múltiples, *aislando* en la medida de lo posible nuestros métodos transaccionales. El aislamiento normalmente involucra el bloqueo de registros o tablas de base de datos, esto se conoce como **locking**.

**Durable:** Sin importar si hay una caída del servidor, una transacción exitosa debe guardarse y *perdurar* posterior al término de una transacción.

En nuestro ejemplo de la compra de boletos, una transacción puede asegurar la **atomicidad** debido a que puede deshacer todos los cambios realizados si alguno de los pasos falla. También se aplicaría la **consistencia** de datos al asegurar que no se deja nada de manera parcial, ejecutando todo o nada. El **aislamiento** prevé que otro usuario tome el boleto reservado mientras todavía no termina la transacción. Y finalmente al momento de hacer commit o rollback de la transacción la información es almacenada sin inconsistencias, permitiendo que sea **durable** incluso posterior a la finalización de la transacción.

# MANEJO DE UNA TRANSACCIÓN



**CURSO JAVA EE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

Al crear aplicaciones empresariales debemos poner especial atención en la persistencia. Además, una aplicación empresarial, según hemos estudiado, se divide en distintas capas, las cuales tienen responsabilidades bien definidas.

En la figura podemos observar en qué momento el objeto Transaction entra en participación. Observamos las 3 capas de una arquitectura empresarial (Cliente – Servicio – Acceso a Datos). En esta arquitectura la transacción comienza en la capa de servicio, la cual es la encargada de establecer la comunicación con la Capa de Datos (DAO).

La capa de datos es la responsable de establecer la comunicación con la base de datos a través del objeto Entity Manager. La transacción comienza desde la capa de servicio, y se propaga a la capa de datos. Esto se debe a que la capa de servicio puede tener comunicación con muchas clases DAO, y por lo tanto la transacción termina hasta que el método de negocio haya insertado, modificado, eliminado y seleccionado todos los datos requeridos de la base de datos, aplicando las características del manejo transaccional que hemos estudiado.

Los métodos de la capa de Servicio, son los que contienen mucha de la lógica de negocio de la aplicación, y por lo tanto es en este nivel donde definimos el manejo transaccional, ya que si lo aplicamos al nivel de la capa de datos, el manejo de commit/rollback por cada operación de un DAO, afectaría las operaciones restantes que tenga un método de negocio.

# CONFIGURACIÓN DE PROPAGACIÓN EN JAVA EE

Demarcación de transacciones por medio de Container-Managed Transactions (CMTs)

Tipo de Propagación	Significado
MANDATORY	El método tiene que ejecutarse dentro de una transacción, de lo contrario se lanzará una excepción.
REQUIRED	El método DEBE ejecutarse dentro de una transacción. Si ya existe una transacción el método la utilizará, de lo contrario creará una nueva.
REQUIRES_NEW	El método DEBE ejecutarse dentro de una transacción. Si ya existe una transacción, se suspende durante la ejecución del método, de lo contrario creará una nueva.
SUPPORTS	Indica que el método no requiere el manejo transaccional, pero puede participar de una transacción si ya hay alguna ejecutándose.
NOT_SUPPORTED	El método NO debe ejecutarse en una transacción. Si ya existe una transacción, se suspenderá hasta la conclusión del método.
NEVER	El método NO debe ejecutarse en una transacción, de lo contrario lanza una excepción.

La propagación indica cómo se comportará un método ante una transacción que ha sido iniciada previamente en otro método, es decir, cómo una transacción se propaga entre métodos transaccionales. El valor por defecto de la propagación es REQUIRED. Los tipos de propagación en una transacción son los siguientes:

- MANDATORY: Si no existe una transacción se lanza una excepción.
- REQUIRED: Este es el comportamiento por default. Si ya existe una transacción, se propaga y la utiliza, de lo contrario crea una nueva.
- REQUIRES\_NEW: Únicamente debe usarse si la acción sobre la base de datos necesita confirmarse (commit) sin importar el resultado de la transacción en curso, por ejemplo, un registro en una bitácora de actividades, por cada intento de persistencia de información, sin importar si se persiste exitosamente o no.
- SUPPORTS: No necesita de una transacción, sin embargo si existe una la utiliza.
- NOT\_SUPPORTED: Debido a que indica que no soporta transacciones, si hubiera alguna, la suspende hasta terminar la ejecución del método marcado con este atributo.
- PROPAGATION\_NEVER : Lanza una excepción si está ejecutándose una transacción.

Para indicar si un método maneja un método distinto de programación se utiliza el código antes de la definición del método:

```
@TransactionAttribute(TransactionAttributeType.SUPPORTS)
```

## EJERCICIO – CURSO JAVA EE

Ejemplo de código de transacciones administradas por el contenedor (CMT):

```

@Stateless
public class PersonaServiceImpl {
    @Resource
    private SessionContext contexto;
    public void modificarPersona(Persona persona) {
        try {
            personaDao.updatePersona(persona);
        } catch (Throwable t) {
            contexto.setRollbackOnly();
        }
    }
    //Más métodos...
}
  
```

**CURSO JAVA EE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

Según hemos visto, el manejo de transacciones se lleva a cabo en los EJB de la capa de servicio o negocio. Como observamos en el código mostrado, el EJB PersonaService contiene métodos de servicio, los cuales por default son transaccionales al ejecutarse en un contenedor empresarial.

El comportamiento por default de un método EJB es de tipo REQUIRED. Sin embargo si queremos modificarlo podemos utilizar el código `TransactionAttribute(TransactionAttributeType.SUPPORTS)` ya sea a nivel de la clase para que afecte a todos los métodos, o al nivel del método que queremos modificar.

Cualquier código que espere arrojar una excepción, y si queremos aplicar rollback de manera explícita deberemos utilizar `setRollbackOnly()` del objeto `SessionContext`, el cual podemos injectar directamente utilizando la anotación `@Resource`.

Sin embargo, utilizar el método `setRollbackOnly` no provocará el rollback de la transacción de manera inmediata, es sólo una indicación de que en cuanto el contenedor termine de ejecutar el método, deberá realizarse el rollback.

Cada transacción JTA está asociada con la ejecución de un hilo (Thread), así que solo se puede ejecutar una transacción a la vez. De tal manera que si una transacción se encuentra activa, el usuario NO puede iniciar otra dentro del mismo hilo (a menos que dicha transacción sea suspendida).

## EJERCICIO – CURSO JAVA EE

- **ABRIR LOS ARCHIVOS DE EJERCICIOS EN PDF.**
- **EJERCICIO:** Manejo Transaccional con Java EE



Experiencia y Conocimiento para tu vida

**CURSO JAVA EE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

**CURSO ONLINE**

# JAVA EMPRESARIAL JAVA EE

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

**CURSO JAVA EE**[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>✓ Programación con Java</li><li>✓ Fundamentos de Java</li><li>✓ Programación con Java</li><li>✓ Java con JDBC</li><li>✓ HTML, CSS y JavaScript</li><li>✓ Servlets y JSP's</li><li>✓ Struts Framework</li></ul> | <ul style="list-style-type: none"><li>✓ Hibernate Framework &amp; JPA</li><li>✓ Spring Framework</li><li>✓ JavaServer Faces</li><li>✓ Java EE (EJB, JPA y Web Services)</li><li>✓ JBoss Administration</li><li>✓ Android con Java</li><li>✓ HTML5 y CSS3</li></ul> |
|--|--|

**Datos de Contacto:**Sitio Web: [www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)Email: [informes@globalmentoring.com.mx](mailto:informes@globalmentoring.com.mx)