

CURSO HIBERNATE & JPA

CICLO VIDA HIBERNATE/JPA



Ing. Ubaldo Acosta

Por el experto: Ing. Ubaldo Acosta



CURSO HIBERNATE & JPA

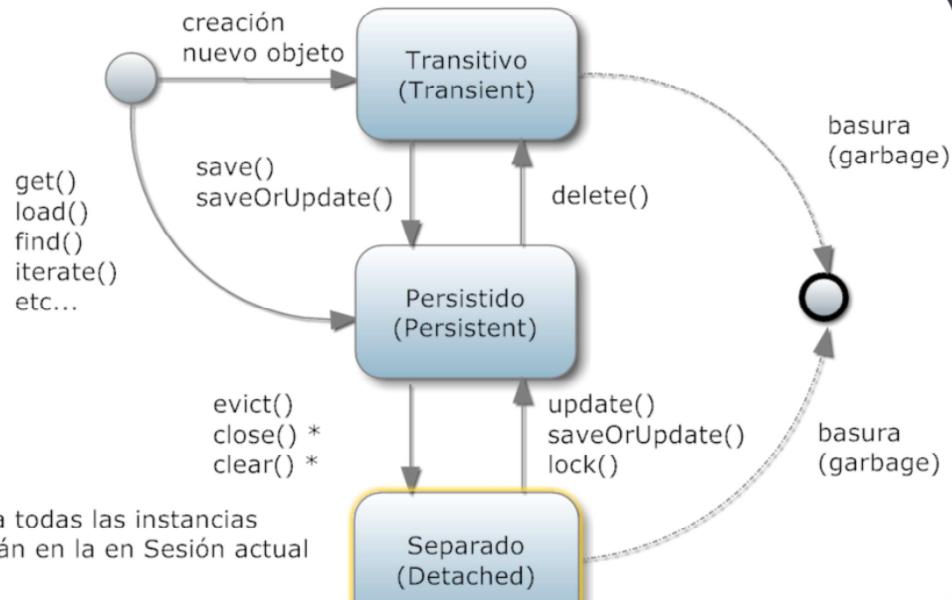
www.globalmentoring.com.mx

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

Vamos a estudiar el tema de Ciclo de Vida en Hibernate y JPA.

¿Estás listo? ¡Vamos!

CICLO VIDA OBJETOS DE ENTIDAD



En esta lección vamos a revisar el tema del ciclo de vida en objetos de entidad.

Los objetos de entidad, según hemos comentado, son clases Java las cuales en el caso de Hibernate o JPA están mapeadas con una tabla de base de datos. Vamos a observar los estados por los cuales pueden pasar estos objetos de entidad.

Observamos en primer lugar que podemos crear un objeto. Cuando creamos un objeto de entidad, por ejemplo un objeto de tipo Domicilio, se crea por medio del operador new, pero esto no quiere decir que de manera automática se persista o guarde en la base de datos, si no que pasa por diferentes estados.

El primer estado se conoce como estado **Transitivo (transient)**, posteriormente ya que inicializamos una transacción podemos ejecutar el método `save()` o si ya existe anteriormente este objeto en la base de datos, también podemos ejecutar el método `saveOrUpdate()` (este método no existe en JPA, por lo que el agregar y actualizar se realizan por separado).

Posteriormente que hemos ejecutado ya estos métodos, pasamos a un Estado de **Persistente (persistent)**, en este estado la transacción todavía está abierta, pero una vez que la transacción se cierra al llamar el método `close()` el objeto de entidad pasa a un estado conocido como **Detached**, es decir, se encuentra separado de la base de datos y la sincronización con nuestra base de datos ya nos se puede garantizar, por lo tanto si hacemos alguna modificación a nuestros objetos en este estado necesitamos volver a sincronizar con la base de datos ya sea por alguno de los métodos `update()`, `saveOrUpdate()` o `merge()`, este último es particularmente utilizado para resincronizar un objeto en estado detached con la base de datos..

En el diagrama observamos los estados por los cuales pasan los objetos Java, pero también observamos cuáles son los métodos que nos permiten cambiar de un estado a otro. Por ejemplo, si nuestro objeto no es nuevo, si no que ya lo tenemos almacenado en la base de datos, podemos recuperarlo por medio de los métodos `get()`, `load()`, `find()`, `iterate()`, etc... y esto nos llevará al estado de persistente, y una vez que cerramos la sesión pasamos nuevamente al estado de detached o separado.

Podemos observar en la imagen que tenemos un asterisco en ciertos métodos debido a que estos métodos `close()` y `clear()` afecta a todas las instancias que se encuentran en la sesión actual, por ello si se cierra la sesión los objetos pasan al estado de **detached**.

En caso de que un objeto en estado de transitivo no se guarde en la base de datos, si no que se deje únicamente en memoria va a pasar a un estado de recolección de basura, debido a que nunca se almacena en la base de datos y por lo tanto se va a perder esta información y el recolector de basura puede eliminar el objeto transitivo.

De igual manera, un objeto detached modificado, que no se volvió a sincronizar con la base de datos, esos cambios no se van a reflejar en la base y el objeto se va a ir directamente al recolector de basura, lo que se va a seguir manteniendo es la información que se sincronizó en su momento, pero los cambios que se hicieron en el estado de detached es decir fuera de la transacción, si no se vuelven a sincronizar, se van a perder debido a que no se sincronizaron nunca con la base de datos. Estos son los estados que tenemos en Hibernate y son muy similares a JPA, y vamos a revisar a continuación más detalle acerca de estos estados.

ALGUNOS ESTADOS DE LOS OBJETOS DE ENTIDAD

Estado Transitivo (Transient):

- ✓ Los objetos de entidad nuevos NO son guardados directamente en la Base de Datos (BD).
- ✓ No están asociados con un registro de BD.
- ✓ Se consideran NO transaccionales.

Estado Persistente (Persistent):

- ✓ Un objeto persistente tiene asociado un registro en la BD.
- ✓ Los objetos persistentes siempre están asociados con una Sesión y son transaccionales. Su estado se sincroniza con la BD al terminar la transacción.

Estado Separado (Detached):

- ✓ Estos objetos tienen asociado un registro de BD, pero su estado no está sincronizado con la BD
- ✓ Todos los objetos recuperados en una transacción se convierten en detached una vez que termina la transacción

CURSO HIBERNATE & JPA

www.globalmentoring.com.mx

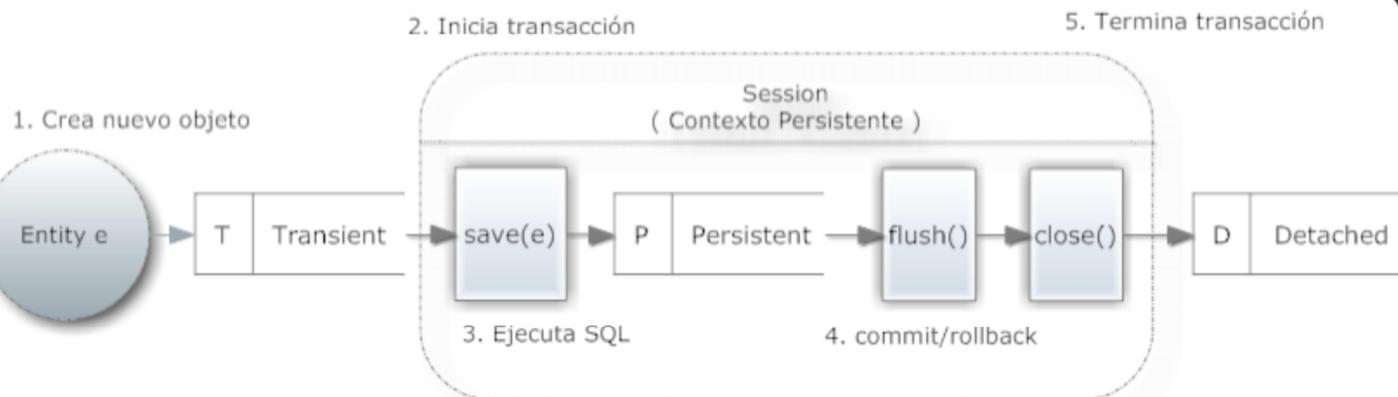
En el estado de **Transitivo** los objetos de entidad nos son guardados directamente en la base de datos. Estos objetos todavía no representan un registro de la base de datos debido a que aún no se sincronizan con la misma, y tampoco se consideran transaccionales debido a que en este estado todavía no existe una transacción.

En nuestro siguiente estado, un objeto **persistent** tiene asociado un registro en la base de datos, por lo tanto si tenemos un objeto el cual ya se encuentra representado en una tabla de base de datos, ya debemos tener asociado la llave primaria en nuestro objeto de Java. También los objetos persistentes siempre están asociados con una sesión o transacción y su estado se sincroniza con la base de datos al terminar dicha transacción, por lo que los objetos en estado de Persistent se van a sincronizar siempre con la base de datos y cualquier modificación que hagamos al objeto se reflejará en la base de datos al finalizar la transacción.

Cuando una transacción ya ha sido finalizada, los objetos pasan a un estado de Detached, estos objetos tienen asociado un registro en la base de datos, pero su estado no está sincronizado con la misma y esto implica que cualquier modificación a un objeto en estado de Detached es necesario sincronizarlo nuevamente con la base de datos y por lo tanto reintegrarlo a una transacción, si no integramos nuevamente este objeto a una transacción los valores que estemos modificando en este objeto Java se van a perder una vez finalizado este proceso.

La diferencia entre el estado transitivo y el estado detached es que el estado transitivo no tiene todavía un registro asociado en la base de datos y el estado de detached ya tiene asociado un registro, pero cualquier cambio en el objeto de Java no se sincroniza de manera automática y finalmente como resumen todos los objetos que se encuentra en una transacción se van a persistir y sincronizar de manera automática al finalizar dicha transacción.

PERSISTIR UN OBJETO EN HIBERNATE/JPA



CURSO HIBERNATE & JPA

www.globalmentoring.com.mx

Vamos a explicar en las siguientes imágenes, los diferentes estados por los cuales puede pasar un objeto en Hibernate o JPA. Acá podemos observar como persistir un objeto en Hibernate o JPA. Lo primero que tenemos que hacer es crear un nuevo objeto de Entidad, por ejemplo, un nuevo objeto de *Domicilio*, se considera que este objeto debido a que todavía no se guarda en una base de datos está en un estado de **transitivo**.

Posteriormente iniciamos una transacción, y todo lo que sucede dentro de este bloque se va a sincronizar con la base de datos. Para crear un nuevo objeto y poderlo almacenar en nuestra base de datos según hemos, tenemos que mandar a llamar alguno de estos métodos para cambiar el estado de nuestro objeto y que pase de transitivo a un estado de persistente, por lo tanto mandamos a llamar el método **save()** y este método lo que va hacer es ejecutar la sentencia **insert** para que este objeto se persista en la base de datos, posteriormente guardamos los cambios haciendo un **commit** de la transacción o llamar al método **flush()**.

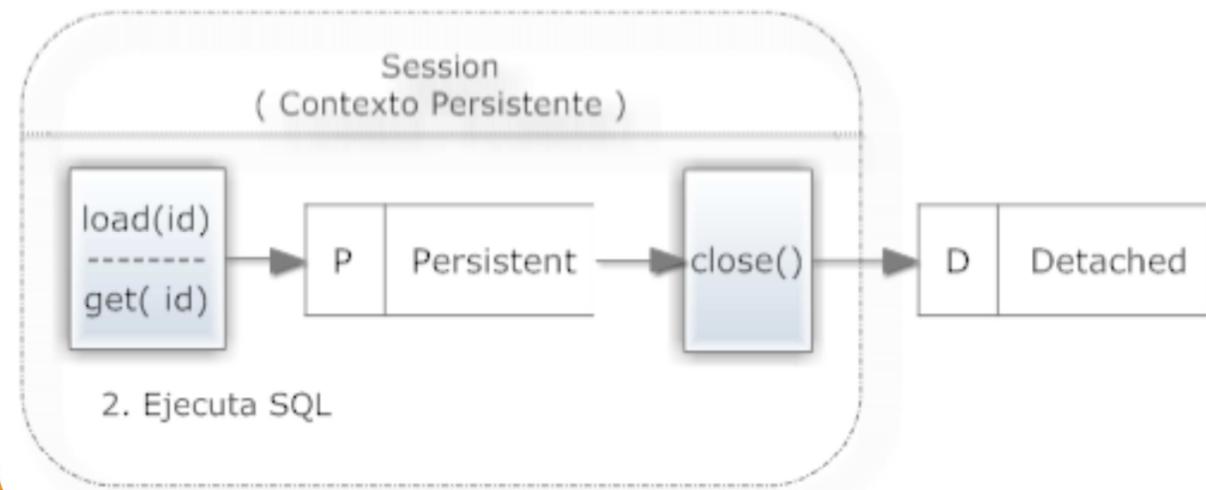
El método **commit** manda a llamar indirectamente el método **flush**. El método **flush** lo que significa es que cualquier SQL pendiente en la transacción actual se va a finalizar y se van a guardar los cambios en la base de datos y finalmente lo que hacemos es terminar nuestra transacción por medio del método **close()**, con esto se termina la transacción y nuestro objeto pasa a un estado de **detached**.

Esto quiere decir que este objeto que hemos creado ya tiene una representación en la base de datos, pero las modificaciones que hagamos en adelante no se van a sincronizar más de manera automática con la base de datos. Ese es el proceso general para persistir un objeto en Hibernate o JPA.

RECUPERAR UN OBJETO EN HIBERNATE/JPA

1. Inicia transacción

3. Termina transacción



2. Ejecuta SQL

CURSO HIBERNATE & JPA

www.globalmentoring.com.mx

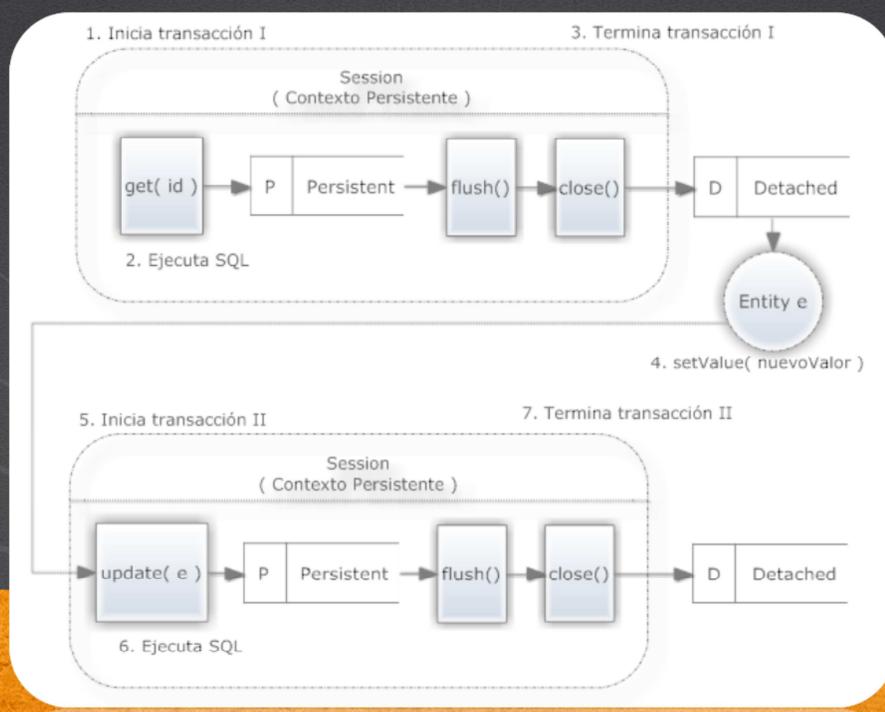
Vamos a revisar a continuación el flujo para recuperar un objeto ya persistido en Hibernate o JPA.

Lo primero que tenemos que hacer es iniciar una transacción y en esa transacción tenemos varios métodos para poder recuperar un objeto que ya se encuentra en la base de datos. Un método se llama load, otro método se llama get, y un tercero se llama merge. Debemos proporcionar ya sea el id o el objeto con el id o llave primaria a recuperar. La diferencia entre estos estos métodos es que el método load si no encuentra la llave primaria que le estamos buscando va a lanzar una excepción y el método get si no encuentra la llave primaria que estamos buscando simplemente deja el objeto como nulo.

Si encontramos el objeto en la base de datos y debido a que todavía estamos en una transacción, el objeto que hemos recuperado pasa al estado de persistente y una vez que cerramos nuestra transacción el objeto pasa a un estado de detached. En este caso los objetos detached los podemos mandar, por ejemplo, a un formulario en la capa de presentación y una vez que hemos terminado de modificar el objeto en estado de detached, tenemos que integrarlo nuevamente a una transacción para poder guardar los cambios realizados por el usuario. Más adelante vamos a ver cómo hacer esto, pero esta es la idea general para recuperar un objeto que tiene una representación en nuestra base de datos.

También existe el método merge, el cual se utiliza cuando ya tenemos una transacción abierta, y entonces es posible pasar un objeto al estado de transitivo o detached al estado de persistente utilizando este método merge dentro de una transacción existente. Más adelante veremos a más detalle como utilizar estos métodos.

MODIFICAR UN OBJETO EN HIBERNATE/JPA



Vamos a revisar ahora como modificar un objeto persistente. Lo que tenemos que hacer es iniciar una transacción y apoyándonos en el flujo anterior, debemos recuperar el objeto a modificar. Vamos a proporcionar una llave primaria del objeto que estamos buscando y una vez que hemos terminado de recuperar el objeto, hacemos commit de la transacción.

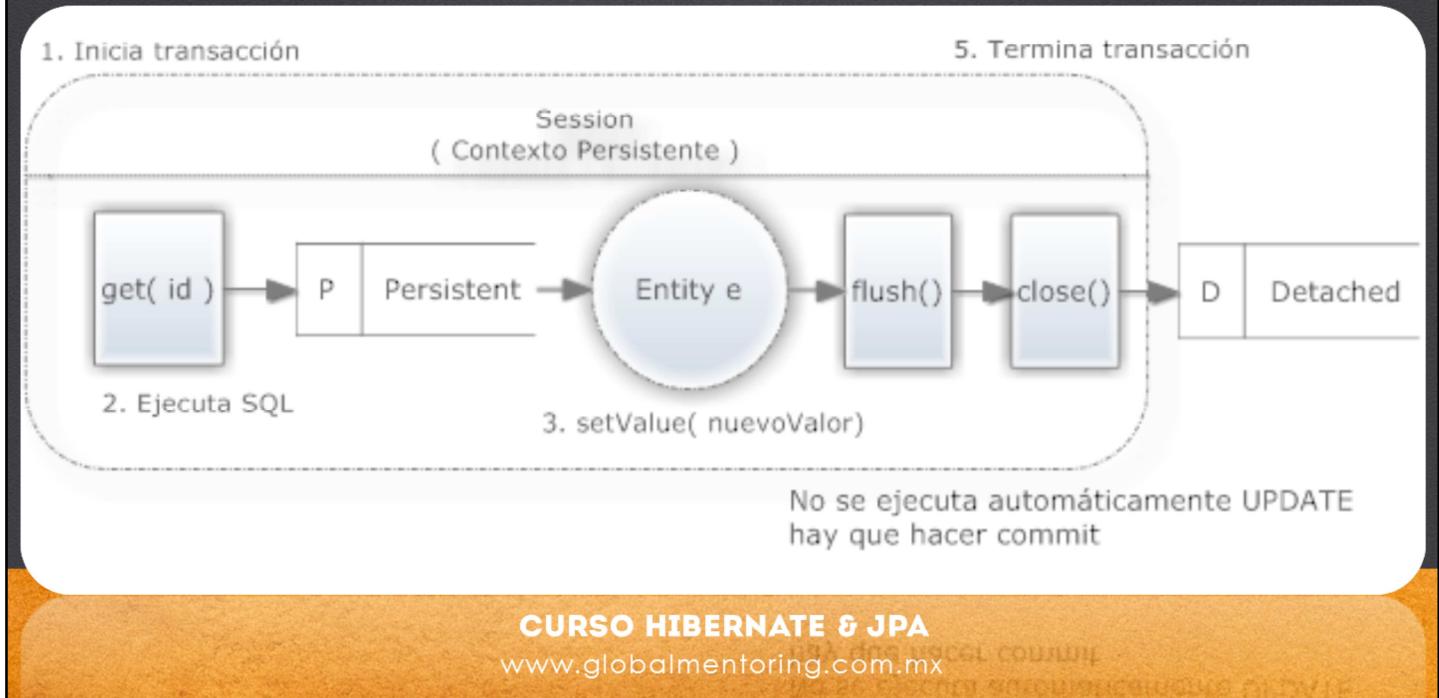
Una vez que hemos terminado la transacción pasamos al estado de detached, es decir, que ya tenemos nuestro objeto de Entidad, pero en este caso no se sincroniza de manera automática, así que los valores que modifiquemos de nuestro objeto por medio de llamadas a los métodos `setValue` no se van a sincronizarán de manera automática con la base debido a que ya hemos terminado la transacción.

Por lo tanto, lo que tenemos que hacer es abrir una nueva transacción. El objeto que ya hemos obtenido anteriormente y que hemos modificado podemos utilizar el método `update` o `merge` para que cambiemos el estado del objeto a persistente. Con esto ya hemos hecho las modificaciones respectivas sobre nuestro objeto de Entidad utilizando una segunda transacción.

Hacemos commit o rollback de la transacción. Si hacemos commit es indirectamente un flush, esto quiere decir que se ha sincronizado los valores que hayamos modificado nuevamente con la base de datos. Finalmente mandamos a llamar el método `close` y nuestro objeto pasa nuevamente al estado de detached pero los valores que modificamos anteriormente ya los hemos guardado de nueva cuenta en la base de datos.

Esto es un ejemplo común cuando recuperamos un objeto de la base de datos, lo modificamos desde la capa de presentación y posteriormente lo enviamos de nueva cuenta a la capa de datos y en la capa de datos nos encargamos nuevamente de guardar la información sincronizando nuestro objeto con la base de datos en una nueva transacción.

MODIFICAR UN OBJETO SESSION LARGA



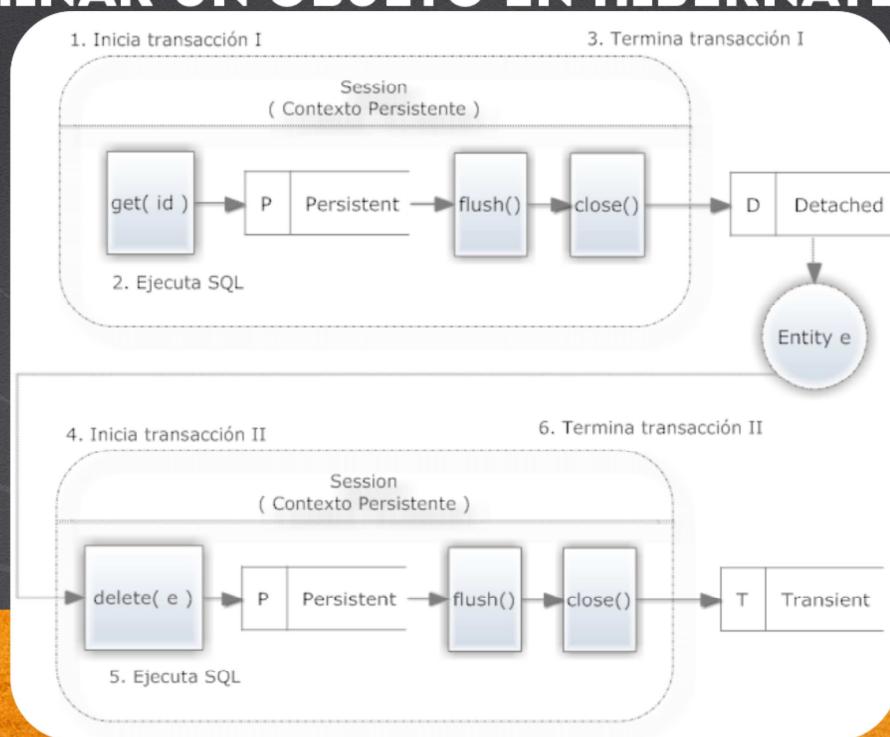
Existe otra manera de modificar nuestro objeto ya persistido en Hibernate o JPA.

Si tenemos una transacción de larga duración lo que podemos hacer es iniciar la transacción y mantenerla abierta lo más posible de tal manera que cuando hayamos recuperado nuestro objeto, por medio del método `get` pasa a un estado de persistido y dentro de esa misma transacción vamos a modificar los valores de nuestro objeto.

Si todavía no hemos terminado la transacción al hacer `commit` indirectamente mandamos a llamar el método `flush`, por lo tanto se van a sincronizar estos cambios con el registro de la base de datos. Finalmente cerramos la transacción y nuestro objeto pasa a un estado de `detached`, por lo que podemos observar en este caso que los valores que hayamos modificado a esta entidad por medio de los métodos `setter` respectivos debido a que todavía estamos dentro de la transacción se van a sincronizar con la base de datos.

Una transacción también se le conoce como el contexto persistente, por lo tanto todas las sentencias SQL que ejecutemos dentro de nuestro contexto persistente si hacemos un `commit` y un `rollback` antes de terminar nuestra transacción se van a sincronizar con nuestra base de datos, por lo tanto todo este bloque se encuentra dentro de un contexto persistente.

ELIMINAR UN OBJETO EN HIBERNATE/JPA



Finalmente vamos a revisar el estado para eliminar un objeto persistente.

En este caso vamos a hacer algo muy similar cuando hicimos la modificación de nuestro objeto, el primero bloque lo que se encarga es de recuperar el objeto que estamos buscando y por lo tanto utilizando el método get proporcionando un identificador. Hacemos los pasos respectivos pasando por el estado persistente, hacemos el flush y cerramos nuestra transacción.

Esto nos va a regresar un objeto en Java y nos lo va a dejar en estado de detached. A este objeto ya no queremos modificarlo, sino eliminarlo de la base de datos, por lo tanto iniciamos una nueva transacción y mandamos a llamar el método delete y le pasamos como parámetro el objeto que queremos eliminar.

Esto cambia el estado del objeto a persistente y al hacer el flush o commit, el objeto se va a eliminar el registro de la base de datos. Esto no implica que el objeto de Java se haya eliminado, y por lo tanto al ya no tener este objeto una representación en la base de datos pasa de nuevo cuenta al estado de transitivo.

Si recordamos el estado de transitivo significa que es un objeto en memoria, pero que no tiene representación en la base de datos, y debido a que lo hemos eliminado de la base de datos, nuevamente tenemos un objeto en estado de transitivo. Aquí la diferencia de un estado de transitivo cuando es nuevo o cuando es eliminado, es que cuando es nuevo la intención es guardarla en una base de datos, pero cuando un objeto termina en estado de transitivo debido a que se eliminó de la base de datos ya no es posible nuevamente guardarlo en la base de datos y esta es una de las diferencias cuando tenemos objetos en estado de transitivo pero que provienen ya sea de una nueva creación de un objeto de entidad o cuando un objeto transitivo se eliminó de la base de datos.

A continuación vamos a crear varios ejercicios para poner en práctica el código respectivo de cada uno de los estados de nuestros objetos de entidad en Hibernate o JPA.

EJERCICIO – CURSO HIBERNATE & JPA

- **ABRIR LOS ARCHIVOS DE EJERCICIOS EN PDF.**
- **EJERCICIO:** Ciclo de Vida con Hibernate/JPA.



Experiencia y Conocimiento para tu vida

CURSO HIBERNATE & JPA

www.globalmentoring.com.mx

CURSO ONLINE

HIBERNATE & JPA

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO HIBERNATE & JPA

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

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx

Email: informes@globalmentoring.com.mx

