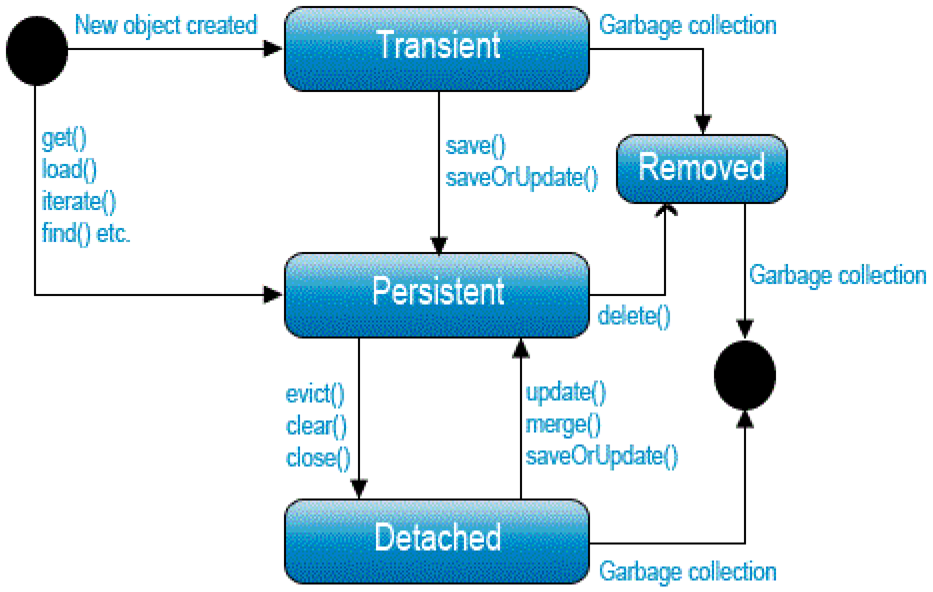
Hibernate Life cycle

Hibernate Life cycle shows how your persistent object managed. Hibernate is ORM based java technology.

Hibernate saves, update and delete record in the database table regarding to value of object as your persistent class is associated with the database table.

Hibernate object life cycle contains four states. These are -

1. Transient
2. Persistent
3. Removed
4. Detached



Transient object s

They get garbage collected when they are not referenced to. Not effected by commits and rollback A transient object becomes persistent if

1) Invoking save on the session object

2) Being referenced by a persistent object

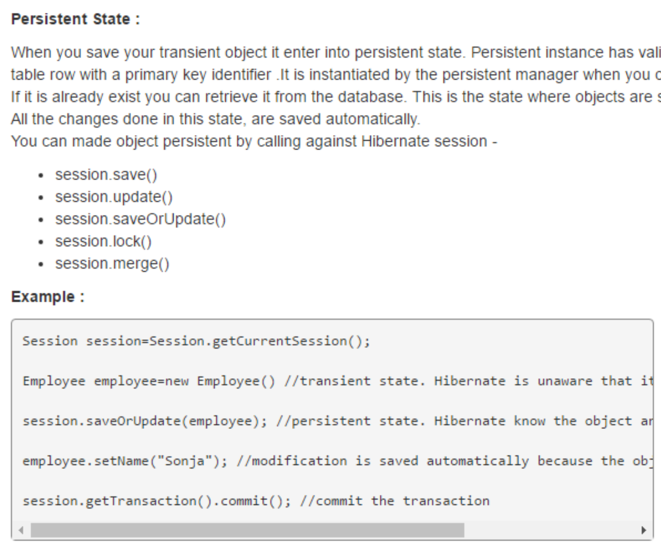
Employee employee=new Employee(); //employee is a transient object

Persistent State :

When you save your transient object it enter into persistent state. Persistent instance has valid database
table row with a primary key identifier .It is instantiated by the persistent manager when you call save().
If it is already exist you can retrieve it from the database. This is the state where objects are saved to the database.

All the changes done in this state, are saved automatically.
You can made object persistent by calling against Hibernate session -

- session.save()
- session.update()
- session.saveOrUpdate()
- session.lock()
- session.merge()



Persistent State :

When you save your transient object it enter into persistent state. Persistent instance has valid database
table row with a primary key identifier .It is instantiated by the persistent manager when you call save().
If it is already exist you can retrieve it from the database. This is the state where objects are saved to the database.
All the changes done in this state, are saved automatically.
You can made object persistent by calling against Hibernate session -

- session.save()
- session.update()
- session.saveOrUpdate()
- session.lock()
- session.merge()

Example :

```
Session session=Session.getCurrentSession();

Employee employee=new Employee() //transient state. Hibernate is unaware that it

session.saveOrUpdate(employee); //persistent state. Hibernate know the object ar

employee.setName("Sonja"); //modification is saved automatically because the obj

session.getTransaction().commit(); //commit the transaction
```

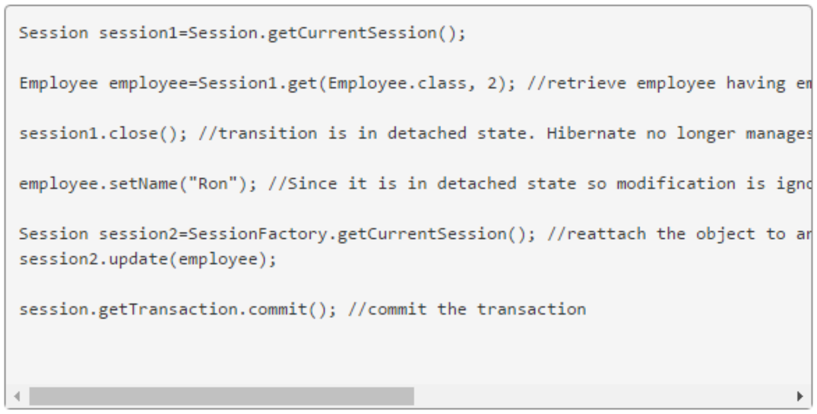
Detached State :

In this state, the persistent object is still exist after closure of the active session.
You can say detached object remains after completion of transaction .It is still represents to the valid row in the database.
In detached state, changes done by detached objects are not saved to the database.
you can not detach your persistent object explicitly. evict() method is used to detach the object from session cache.
clear() ,close() methods of session can be used to detach the persistent object.

For returning in to the persistent state, you can reattach detached object by calling methods -

- update()
- merge()
- saveOrUpdate()
- lock() - It is reattached but not saved.

Example :



```
Session session1=Session.getCurrentSession();

Employee employee=Session1.get(Employee.class, 2); //retrieve employee having em

session1.close(); //transition is in detached state. Hibernate no longer manage

employee.setName("Ron"); //Since it is in detached state so modification is ign

Session session2=SessionFactory.getCurrentSession(); //reattach the object to a
session2.update(employee);

session.getTransaction.commit(); //commit the transaction
```

Removed State :

When the persistent object is deleted from the database ,it is reached into the removed state.
session.delete()
At this state java instance exist but any changes made to the object are not saved to the database.
It is ignored by Hibernate and when it is out of scope, it is assigned to garbage collection.

Example :

```
Session session=Session.getCurrentSession();



Employee employee=Session.get(Employee.class, 2); //retrieve employee having empId 2.
employee return persistent object.



session.delete(employee); //transition is in removed state. database record is delete
d by Hibernate and no longer manages the object.



employee.setName("Ron"); //Since it is in removed state so no modification done by Hi
bernate

session.getTransaction.commit(); //commit the transaction
```