Hibernate Lifecycle

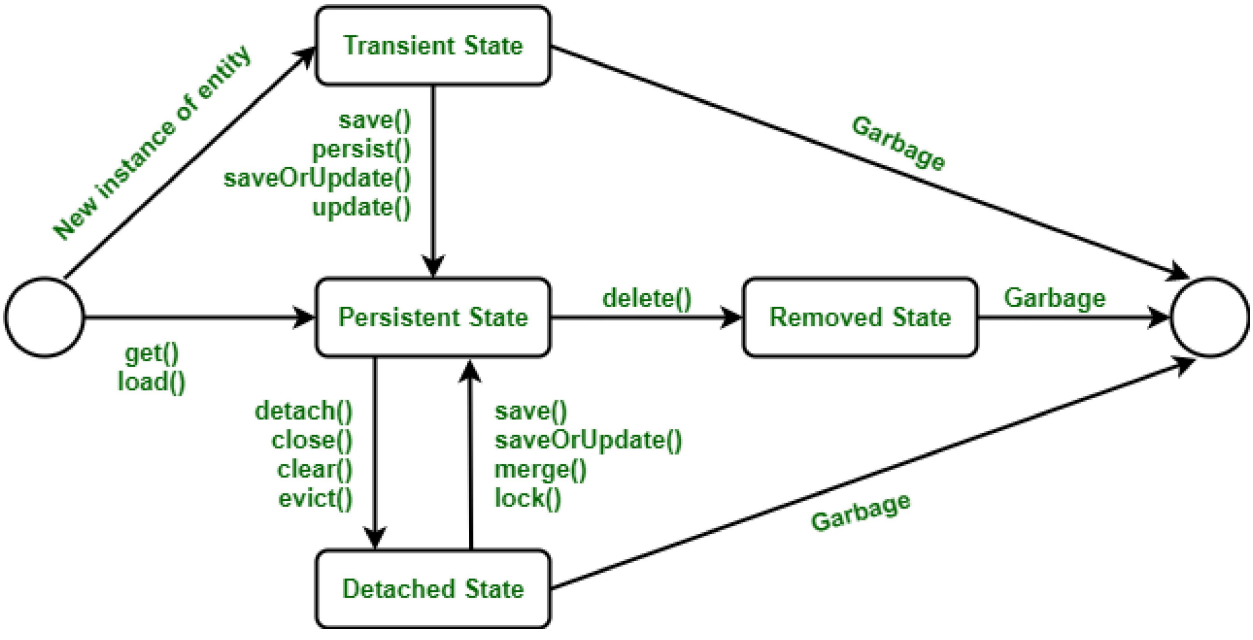
Last Updated : 23 Jul, 2025

In this article, we will learn about **Hibernate Lifecycle,** or in other words, we can say that we will learn about the **lifecycle of the mapped instances of the entity/object classes in hibernate**. In Hibernate, we can either create a new object of an entity and store it into the database, or we can fetch the existing data of an entity from the database. This entity is connected with the lifecycle and each object of the entity passes through the various stages of the lifecycle.

There are mainly **four main states** of the Hibernate Lifecycle :

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

Hibernate Lifecycle Diagram

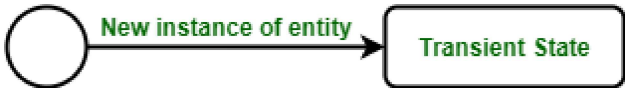


Hibernate Lifecycle

As depicted from the above image, one can co-relate how they are plotted to plot better in our mind. Now we will be discussing the states to better interpret Hibernate lifecycle. It is as follows:

State 1: Transient State

The transient state is the first state of an entity object. When we instantiate an object of a POJO class using the new operator then the object is in the transient state. This object is not connected with any hibernate session. As it is not connected to any Hibernate Session, So this state is not connected to any database table. So, if we make any changes in the data of the POJO Class then the database table is not altered. Transient objects are independent of Hibernate, and they exist in the **heap memory**.



Changing new object to Transient State

How Transient State Occurs:

There are two layouts in which transient state will occur as follows:

- When objects are generated by an application but are not connected to any session.
- The objects are generated by a closed session.

Here, we are creating a new object for the Employee class. Below is the code which shows the initialization of the Employee object

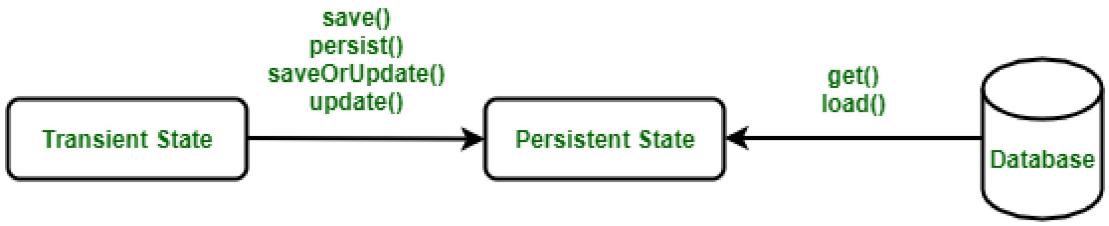
```
//Here, The object arrives in the transient state.
Employee e = new Employee();
e.setId(21);
e.setFirstName("Neha");
e.setMiddleName("Shri");
e.setLastName("Rudra");
```

State 2: Persistent State

Once the object is connected with the Hibernate Session then the object moves into the Persistent State. So, there are two ways to convert the Transient State to the Persistent State :

- Using the hibernated session, save the entity object into the database table.
- Using the hibernated session, load the entity object into the database table.

In this state. each object represents one row in the database table. Therefore, if we make any changes in the data then hibernate will detect these changes and make changes in the database table.



Converting Transient State to Persistent State

Following are the methods given for the persistent state:

- session.persist(e);
- session.save(e);
- session.saveOrUpdate(e);
- session.update(e);
- session.merge(e);
- session.lock(e);

Example:

```
// Transient State
Employee e = new Employee("Neha Shri Rudra", 21, 180103);
```

```
// Persistent State
session.save(e);  // The object is now persistent and saved to the database
```

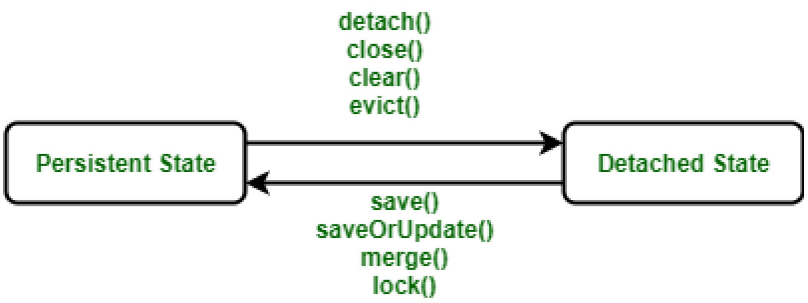
State 3: Detached State

For converting an object from Persistent State to Detached State, we either have to close the session or we have to clear its cache. As the session is closed here or the cache is cleared, then any changes made to the data will not affect the database table. Whenever needed, the detached object can be reconnected to a new hibernate session. To reconnect the detached object to a new hibernate session, we will use the following methods as follows:

- merge()
- update()
- load()
- refresh()
- save()
- update()

Following are the methods used for the detached state :

- session.detach(e);
- session.evict(e);
- session.clear();
- session.close();



Converting Persistent State to Detached State

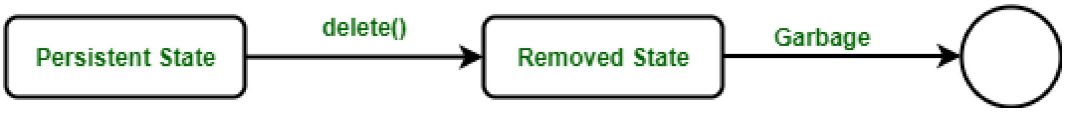
Example:

```
// Transient State
Employee e = new Employee("Neha Shri Rudra", 21, 180103);
// Persistent State
session.save(e);
// Detached State
session.close();  // The object is now detached
```

State 4: Removed State

In the hibernate lifecycle it is the last state. In the removed state, when the entity object is deleted from the database then the entity object is known to be in the removed state. It is done by calling the ***delete() operation***. As the entity object is in the removed state, if any change will be done in the data will not affect the database table.

Note: To make a removed entity object we will call **session.delete()**.



Converting Persistent State to Removed State

Example:

```
// Transient State
Employee e = new Employee();
Session s = sessionFactory.openSession();
e.setId(01);
// Persistent State
session.save(e);
// Removed State
```

```
session.delete(e);  // The object is now removed from the database
```

Comment | More info | Advertise with us

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305