# **Hibernate - Persistent Class**

The entire concept of Hibernate is to take the values from Java class attributes and persist them to a database table. A mapping document/file helps Hibernate in determining how to pull the values from the classes and map them with table and associated fields. ()

Java classes whose objects or instances will be stored in database tables are called persistent classes in Hibernate. Hibernate works best if these classes follow some **simple rules**, also known as the **Plain Old Java Object** (POJO) programming model.

//java class ,POJO-plain old java object

//VO-value object

//BO- business object

//model

//java bean

//Persistent Class

//Entity Class

There are following main rules of persistent classes, however, none of these rules are hard requirements −

* Public class resides inside package.
* All Java classes that will be persisted need a default constructor.
* All classes should contain an ID in order to allow easy identification of your objects within Hibernate and the database. This property maps to the primary key column of a database table.(Id)
* All attributes that will be persisted should be declared private and have **getXXX** and **setXXX** methods defined in the JavaBean style.
* A central feature of Hibernate, proxies, depends upon the persistent class being either non-final, or the implementation of an interface that declares all public methods.

The POJO name is used to emphasize that a given object is an ordinary Java Object, not a special object, and in particular not an Enterprise JavaBean.

## Image result for object life cycle in hibernateHibernate Object Life Cycle?

Hibernate prefers your objects to be in a certain “state”, known as the persistent state… this persistent state is one of four different states that exist inside of the hibernate persistence life cycle. There are four states of Hibernate Object Life Cycle :

1. Transient- When ever an object of a pojo class is created then it will be in the Transient state. One newly created object, without having any relation with the database, means never persistent, not associated with any Session object.
2. Persistent- Having the relation with the database, associated with a unique Session object
3. Detached- previously having relation with the database [persistent ], now not associated with any Session.
4. Removed- A persistent object is considered to be in the removed state when a delete() operation is called on it. Note that Once you've deleted .an object and moved to the “removed” state, you should no longer use that particular object for any reason.

**1. New or Transient State:**

When ever an object of a POJO class is Created(instantiated) using the new operator then it will be in the Transient state; this object is not associated with any Hibernate Session.

For example,

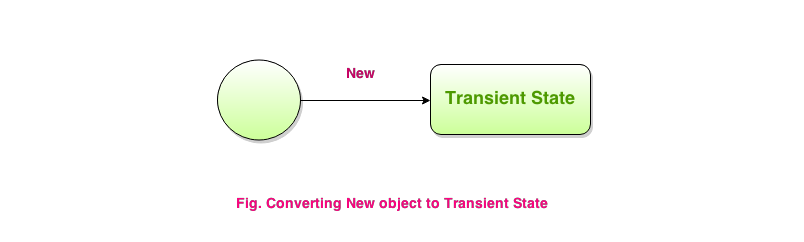
Employee employee = new Employee("Smita", 27, 30998); *// Transient*

When we call delete() on persistent object then it also moves to transient state.

session.delete(employee);

This object don’t have any association with any database table row. In other words any modification in data of transient state object doesn't have any impact on the database table. so their state is lost as soon as they’re no longer referenced by any other object.

**Note: Transient objects exist in heap memory.**

[](http://3.bp.blogspot.com/-Zw9IEqjYEZU/VKNUD6qxN4I/AAAAAAAADwM/_HdRAIaOpEY/s1600/Hibernate+Object+Life+Cycle+1.png)

Transient state will be happened two scenarios:

1. First where the objects are created by application but not connected to a session, and
2. Second the objects are created by a closed session.

**Converting Transient State to Persistence State:**

* By saving the that object
  + session.save()
  + session.persist()
  + session.saveOrUpdate()
* By loading that object from database
  + session.load()
  + session.get(),
  + session.byId()
  + session.byNaturalId() etc..

**2. Persistent or Managed State:**

In order to convert or move an object from Transient to Persistent, there are two ways.

1. Saving the object to the database using session
2. Loading the object from the database using session

In this state object known to the hibernate and represent a one row in the database. Hibernate will detect any changes made to an object in persistent state and synchronize the state with the database when the unit of work completes.

**Different ways to Save an Object:**

Hibernate supports the different ways to save an object to the database. They are

* session.save()
* session.saveOrUpdate()
* session.persist()

For example,

Employee employee = new Employee("Smita", 27, 30998); *// Transient*  
session.save(employee); *// persitent*

**Different Ways to Load an Object:**

Hibernate supports the different ways to load an object from the database. Few of them are

* session.get()
* session.load()
* session.byId()
* session.byNaturalId() etc...

For example,

Employee employee = session.get(Employee.class, 1); *// here employee object is associated with session. So employee object state is Persistent.*

|  |
| --- |
| [http://4.bp.blogspot.com/-4fxjPBz_Pxc/VKNUD2iyPGI/AAAAAAAADwY/hyRkMwS_4Mo/s1600/Hibernate%2BObject%2BLife%2BCycle%2B2.png](http://4.bp.blogspot.com/-4fxjPBz_Pxc/VKNUD2iyPGI/AAAAAAAADwY/hyRkMwS_4Mo/s1600/Hibernate+Object+Life+Cycle+2.png) |
| **Persistent State** |

We can covert the object from persistent state to detached state by clearing the cache of the session or close the session using evict(), clear() and close() methods.  
 **Converting Persistent State to Detached State:**

* session.evict();
* session.clear();
* session.close();

**3. Detached State:**

In order to convert or move an object from Persistence to Detached State, we need to clear the cache of the session or close the session by using following methods.

* session.evict();  - clear particular object from the cache
* session.clear();  - clears all objects from the cache
* session.close();

Employee employee = new Employee("Smita", 27, 30998); *// Transient*  
session.save(employee); *// persitent*  
session.close(); *// here employee state is detached because currently it is not associated with session.*

|  |
| --- |
| [http://4.bp.blogspot.com/-N7VZ0K6em6k/VKNUD3a2_7I/AAAAAAAADwQ/KHEMFR_DoUA/s1600/Hibernate%2BObject%2BLife%2BCycle%2B3.png](http://4.bp.blogspot.com/-N7VZ0K6em6k/VKNUD3a2_7I/AAAAAAAADwQ/KHEMFR_DoUA/s1600/Hibernate+Object+Life+Cycle+3.png) |
| **Detached State** |

Changes made in this object does not reflect to the database. But we can change the state to persistent by calling following methods on detached object.

* session.update()
* session.merge()
* session.saveOrUpdate()

**Converting Detached State to Persistent State:**

* session.update()
* session.merge()
* session.saveOrUpdate()

**4. Removed State:**  
This is last state in the hibernate object life cycle. A persistent object is considered to be in the removed state when a delete() operation is called on it. Note that Once you've deleted an object and moved to the “removed” state, you should no longer use that particular object for any reason.

For example:

session.delete(employee);

|  |
| --- |
| [http://4.bp.blogspot.com/-6xyDX0rTO7Q/VKNUFTNeC4I/AAAAAAAADwg/z_eITXbSwOY/s1600/Hibernate%2BObject%2BLife%2BCycle%2B4.png](http://4.bp.blogspot.com/-6xyDX0rTO7Q/VKNUFTNeC4I/AAAAAAAADwg/z_eITXbSwOY/s1600/Hibernate+Object+Life+Cycle+4.png) |
| **Removed State** |

**Example:**

package com.synergetics.client;  
  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.Transaction;  
  
import com.synergetics.pojo.Employee;  
import com.synergetics.util.HibernateUtil;  
  
public class Application {  
 public static void main(String[] args) {  
   
 SessionFactory sessionFactory = HibernateUtil.getSessionFactory();  
 Session session = sessionFactory.openSession();  
   
 // Hibernate Object Life Cycle  
   
 // New or Transient State begin  
 Employee employee = new Employee("Smita", 27, 30998);   
 // New or Transient State end  
   
 System.out.println("Session Info in Transient State : ");  
 System.out.println(session);  
   
 // Persistent or Managed State begin  
 Transaction transaction = session.beginTransaction();  
 session.save(employee);  
 transaction.commit();  
 // Persistent or Managed State end  
   
 System.out.println("Session Info in Persistent State : ");  
 System.out.println(session);  
   
 // Detached State begin  
 session.evict(employee);  
 // Detached State end  
   
 System.out.println("Session Info in Detached State : ");  
 System.out.println(session);  
   
 // Persistent State begin  
 transaction = session.beginTransaction();  
 employee.setName("Smita Reddy");  
 employee.setAge(27);  
   
 session.saveOrUpdate(employee);  
 transaction.commit();  
 // Persistent State end  
   
 System.out.println("Session Info in Persistent State : ");  
 System.out.println(session);  
   
 // Removed State begin   
 session.delete(employee);  
 // Removed State end   
   
 System.out.println("Session Info in Removed State : ");   
 System.out.println(session);  
   
 sessionFactory.close();  
 }  
   
}