Hibernate Object Life Cycle

Hibernate work like a transformer it takes a Java Object and save this Object as a row of a database table.

So if you look minutely at the java Objects we can observe following four possible transitions

A Normal Java Object.

Java Object associated with Hibernate Framework.

Java Object Persist as a Database row.

Java Object detached from Hibernate Framework.

Based on this transition Hibernate introduces a different kind of logical State and provide helper methods to transit from one state to another. Some cases it is possible to go back and forth some cases it is not. We will discuss it shortly.

**Hibernate Object State:**

Hibernate maintains four States.

1. **Transient :** When you create an Object/POJO, Hibernate considers this POJO as a Transient Object. So any Object you created while developing is a Transient Object to Hibernate. To recognize Transient Objects please use this checklist.

* Transient Object has no representation in DataBase. That means this Object has no corresponding row in a Table. Say we created an Employee Object ( the name is Shamik) and it maps to employee table, so if it is transient then you will not found this employee in the employee table.
* Transient Object has no Identifier/Unique id or Primary key according to RDBMS. It is natural as by point 1 if it has no representation in the database then how it has an identifier.
* Transient Object will not be attached to Hibernate Session Object or in a simple term it is not bound by any transactional context.

So by default ,any Java Objects is Transients to Hibernate and Hibernate not track this Objects as it has no database representation.

2. **Persistence** : The most important state in Hibernate life cycle, when a transient Object persists in Database or a database row is fetched by Hibernate framework as Object/POJO. We call this Object has a Persistence state. So Persistence state has following characteristics.

* Persistence Object must have a representation in DataBase. That means this Object has to map a row to a Table.
* Persistence Object has an Identifier/Unique id or Primary key according to RDBMS.
* Persistence Object always attached to Hibernate Session Object or in a simple term it is enclosed by a transactional context.

3. **Detached** : The detached state is the most difficult to understand. A Detached Object has a representation in RDBMS but it is not associated with any transactional context or Session Object.

So Detached state has following characteristics.

* Detached Object must has a representation in DataBase. That means this Object has to map a row to a Table.
* Detached Object has an Identifier/Unique id or Primary key according to RDBMS.
* Detached Object not attached to Hibernate Session Object or in a simple term it is not associated by a transactional context.

**Hibernate Methods State Transitions:**

|  |  |
| --- | --- |
| **State** | **Hibernate Method** |
| Transient to Persistence | session.save() |
| Persistence State | session.load() |
| Persistence State | session.get(); |
| Persistence State | session.saveorUpdate(); |
| Persistence to Detached | session.close() |
| Detached To Persistence | session.update() |
| Detached To Persistence | Session.merge(); |

**Brief Method Details:**

|  |  |
| --- | --- |
| **Method name** | **Details** |
| save | Used for save a POJO/Transient object into database and attach an identifier with it. |
| load | Used for Lazy loading, when developer tries to load an Object from database it does not hit database rather returns a Proxy Object. Later when developers call any method on that proxy, then actual database calls happens. If the Object is not found in database it throws ObjectNotFoundException |
| get | Used for eager fetching. It always hit the database if Object is not found in database it returns null |
| saveorUpdate | If Object is not found in database it creates a new one and generates an Identifier for it else update the state of the row in database. |
| close | To close the Session. It will flush the session cache and all the queries are applied on tables. |
| update | It will update the entity and save the updates state on database while closing /flushing the session. |
| merge | It also used for update but the main difference from update is after merge it returns a new Object which will copy of Original object and copied the updated values, so it does not touched the original entity, and this Copied entity will be tracked by Hibernate so any further changes on copied entry is being tracked but if any changes does on original entity it is not tracked by Hibernate so that will not reflect in database while closing or flushing the session. |

**State Transition Diagram:**



**State Transition diagram with method name.**