What is Hibernate?

Hibernate is an Object-Relational Mapping (ORM) tool which reduces the difficulty in the application development. Hibernate provides a framework that interacts with the data stored in the databases. It also uses the specifications of the Java Persistence API (JPA) and licensed by GNU Lesser General Public License (LGPL). It supports almost all relational databases.

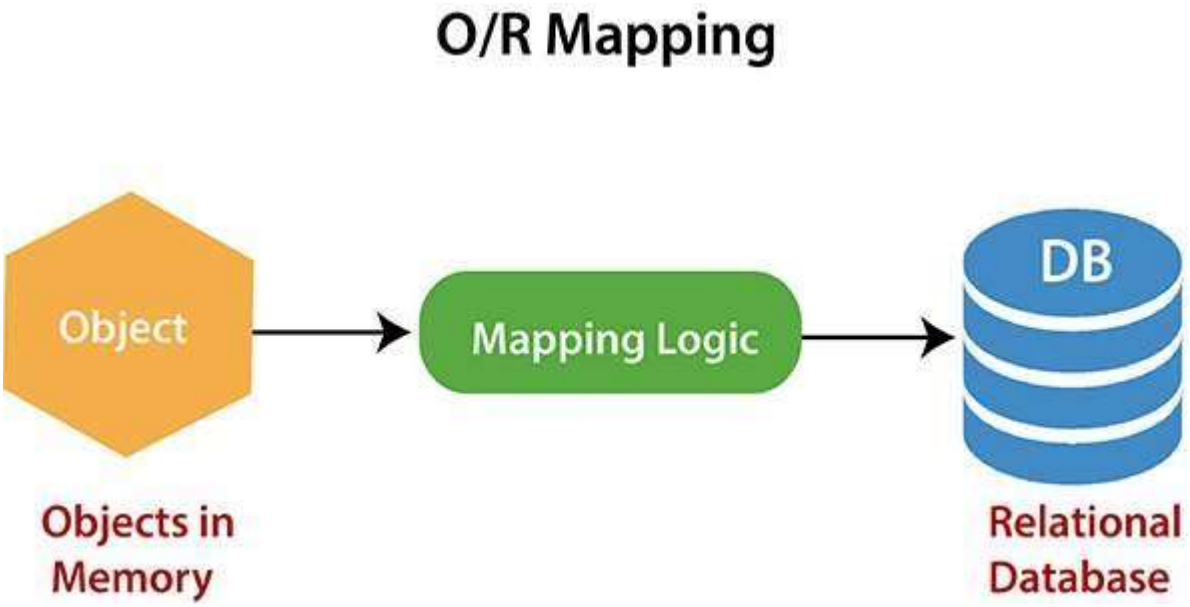
Gavin King is known as the father of hibernate. Hibernate can be used both in Java SE (Standard Edition) and Java EE (Enterprise Edition).
Hibernate is an essential component of JBoss EAP (Enterprise Application Platform). JBoss tools provide a smooth implementation and testing of code in application development.

What is ORM tool?

An **Object-Relational Mapping tool** helps to clarify the creation, manipulation, and access to the data. With this technique, the object is mapped to the data stored in the database. ORM is used to solve some mismatchproblems like

- **Granularity**- It is defined as an object which can further be divided into new data fields. For example, a person as an entity. We can divide 'person' into new more attributes like name, address, contact,etc.

- **Inheritance**- Inheritance enables a sub/child class to inherit the properties and attributes of its super/ parent class.
- **Identity-** A Relational databases provide a small concept of identity. It is referred to as a column that automatically generates numeric values.
- **Association**- It defines a relation between two objects based on a common attribute.
- **Navigation-** It is a database in which records and objects are found through the reference of other objects. The method of object access is different in both Java and RDBMS.



Java Persistence API (JPA)

- JPA defines a set of functionalities, standards, and concepts to the ORM tool. It is available in the javax.persistence package. To decrease the line of codes for relational object management, a coder must follow the "JPA Framework," which easily allows interaction with the database.

- JPA is an open-source framework and was first released on May 2006. There are many enterprise vendors like Oracle, Eclipse, etc. which provide new JPA products.

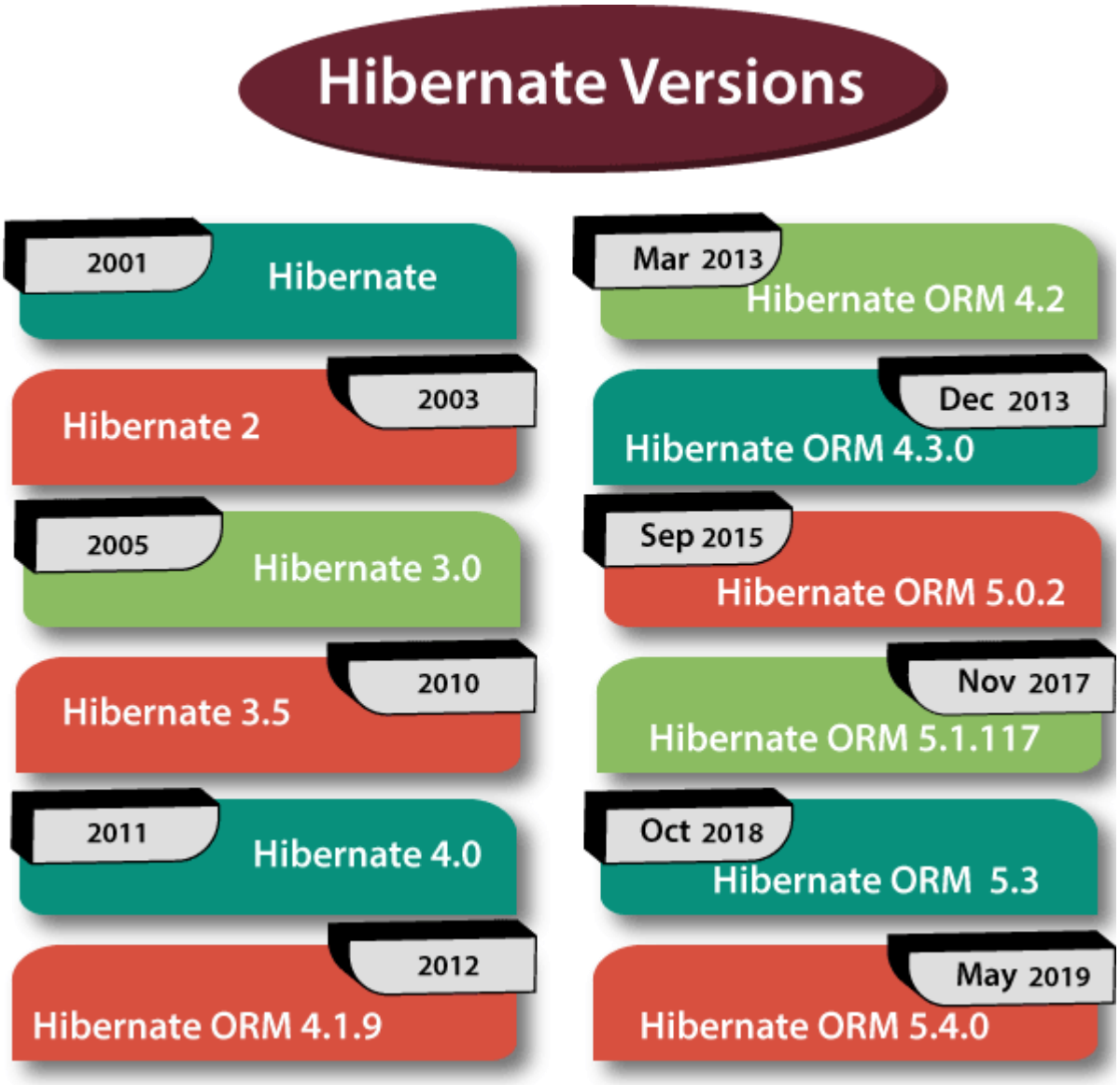
Hibernate History and Versions

History of Hibernate

Hibernate was developed in 2001 by Gavin king with his colleagues from Circus Technologies. The main aim was to provide better persistence capabilities than those of EJB2, by reducing the complexities and adding somemissing features.

Hibernate is an ORM (Object-Relational Mapping) tool which simplifies the application development. Hibernate provides a framework that interacts with the data stored in the database, and it is a framework for Java EE 5. Gavin King has also contributed to the design of EJB 3.0 and JPA.

Following are the versions of Hibernate:

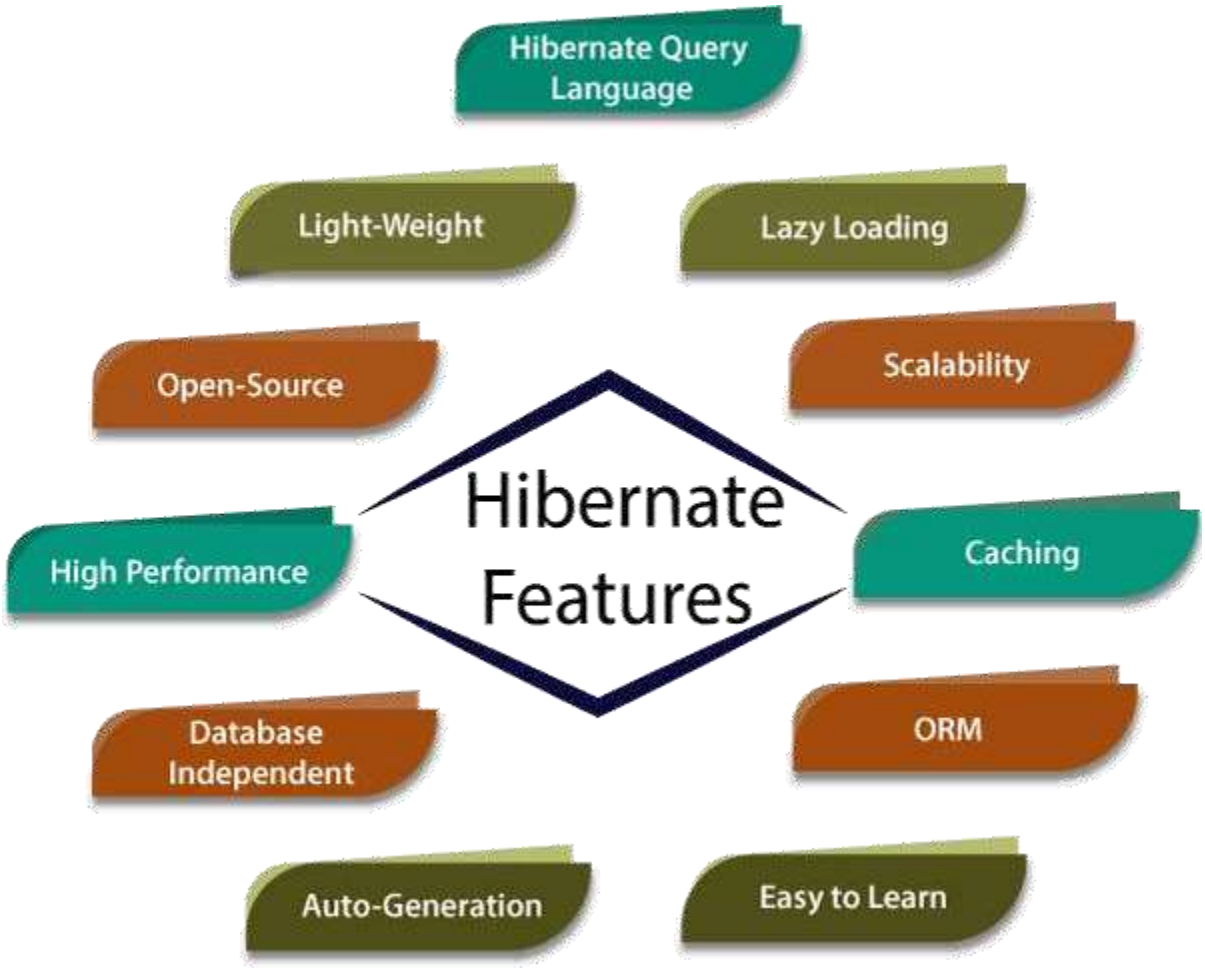


- **2001:** The first version of Hibernate was released. The main aim was to provide better persistence capabilities than those of EJB2, by reducing complexities and adding some missing features.
- **2003:** The new version of Hibernate was launched that is hibernate2, which provides many improvements over the first version.

- **2005:** The third version of Hibernate was released known as hibernate 3.0 with some new key features. Some features are an interceptor, user-defined filters, and Annotations of JDK 5.0.
- **2010:** Hibernate 3 (version 3.5 and up) has become a certified implementation of the JPA 2.0specification.
- **2011:** The new version of Hibernate was released known as Hibernate 4.0.0 with some new features such as support, multi-tenancy, better sessionopening, and many more.
- **2012:** Another version of Hibernate was released known as Hibernate ORM 4.1.9.
- **2013:** New version Hibernate ORM 4.2 Final was released in March 2013. New version Hibernate ORM 4.3.0 Final was released on December 2013 with a new feature JPA 2.1.
- **2015:** New version Hibernate ORM 5.0.2 Final was released on September 2015 with some improvements like bootstrapping, hibernate-java8,Karaf support, etc.
- **2017:** In November 2017, Hibernate ORM 5.1.17 wasreleased.
- **2018:** New version Hibernate ORM 5.3 Final was released in October 2018.
- **2019:** New version Hibernate ORM 5.4.0 Final was released in May 2019.

It runs on a platform known as JVM (Java Virtual Machine). Java Virtual Machine (JVM) is a virtual machine which is used to run Java programs and applications on the computer. Other language programs are also compiled to Java bytecode can be run on JVM. In other words, JVM is a virtual engine that serves the runtime environment to the Java programs and application. JVM converts Java bytecode into machine language.

Hibernate Key Features



1. Lightweight

- Hibernate is a lightweight framework as it does not contains additional functionalities; it uses only those functionalities required for object-relational mapping.
- It is a lightweight framework because it uses persistent classes for data transfer between java application and databases.

2. Open Source

- Hibernate is open-source software that means it is available for everyone without any cost. Its source code is made available so that anyone can use and develop applications using hibernate.

It can be downloaded from its official website, hibernate.org. The latestversion a user can download is Hibernate 6.4.1.Final

-

3. ORM (Object Relation Mapping)

- Hibernate is an ORM tool which helps in the interaction between the java classes and relationaldatabase.
- It also solves the problem of data mismatch found in Java application and RDBMS.

4. High Performance

- Hibernate supports many different fetching techniques such as, caching, lazy initialization, and many more to achieve high performance.

5. HQL (Hibernate Query Language)

- Hibernate has its query language, i.e., HQL (Hibernate Query Language) which is independentof the database.
- HQL is an object-oriented language similar to SQL, but it works with the persistent object and its properties.

6. Caching

- Hibernate supports two levels of caching, first-level and second-level caching.
- Caching is the process of storing data into cache memory and improves the speed of data access.

7. Auto-Generation

- Hibernate provides a feature of automatic table generation. It means a programmer need not worry about the query implementation, i.e., Hibernate does on its own.

8. Scalability

- Hibernate is highly scalable as it can be fit in any environment. Hibernate can be used for both small scale and large scale applications.

9. Lazy Loading

- Hibernate supports a new concept called lazy loading. Lazy loading concept retrieves the only necessary object for execution.
- It also improves the performance of an application.

10. Easy to learn

- Hibernate is easy to learn and implement. It is developer-friendly as it takes care of the changes made to the database automatically, so it reducesthe developer's work.

11. Database Independent

- Hibernate is database-independent as it provides 'Database Dialect' so we need not write SQL queries.
- It supports many databases such as Oracle, MySql,Sybase, etc.

Architecture of Hibernate

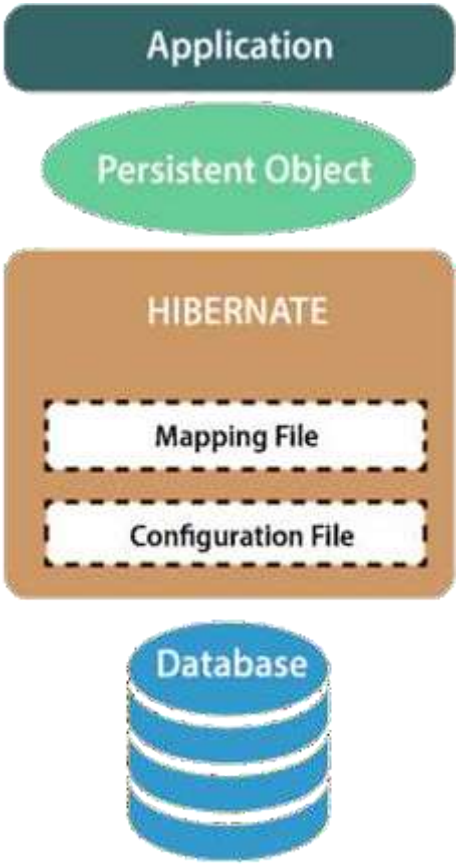
- Hibernate is an ORM framework with a layered architecture. It consists of many objects, such as a

persistent object, sessionfactory object, transaction object, session, etc.

- There are mainly three layers in Hibernate architecture:-

-

- Application Layer
- Hibernate Core Layer
- Database Layer



Core Components of Hibernate

- Before the creation of the first Hibernate application, we must know the core components of Hibernate. They are listed below:-

1. Configuration

The **org.hibernate.cfg** package contains the Configuration class, which consists of the properties and function files of Hibernate. The configuration object is created only once during the application initialization.
To activate the Hibernate Framework, we use the following code:

Configuration cfg=new Configuration();

It reads both mapping and configuration file.

cfg.configure();

2. SessionFactory

The **org.hibernate.sessionFactory** package contains the SessionFactory interface whose object can be obtained by the object of Configuration class. It is a threadsafe object and used by all the threads in the application.
The below code shows the creation of SessionFactory object:
SessionFactory factory=cfg.buildSessionFactory();

It takes the JDBC information from cfg object and creates a JDBC connection.

3. Session

The **org.hibernate.session** package contains the Session interface. A SessionFactory object is used to create a Session object, which is lightweight object. The Session object is not threadsafe. It is used to execute CRUD

operations (insert, delete, update, edit). It also holds the first-level cache data in Hibernate.

The below code shows the creation of Session object:

Session session=factory.buildSession();

4. Transaction

The org.hibernate.transaction package contains a Transaction interface. The object of the session creates a Transaction object. It provides the instruction to the database for transaction management. It is a short-lived single-threaded object.

The below code shows the creation of Transaction object:

Transaction t=session.beginTransaction();t.commit();

The **commit()** function is used to close the transaction.

5. Query and Criteria

The **org.hibernate.query** and **org.hibernate.criteria** package contains Query and Criteria interface, respectively. Query objects use Hibernate Query Language (HQL) to get data from the database. Criteria objects are also used to retrieve data from databases. Session objects are used to create query and criteria objects.

The below code shows the initialization of Query object: **Query query= session.createQuery();**

The below code shows the Initialization of criteria object: **Criteria criteria=session.createCriteria();**

Hibernate SessionFactory

A **SessionFactory** is an interface in Hibernate that create the instances of Session interface. It is a heavy weight object, and it is usually created during the application startup and kept for later use. It is a **threadsafe** object and used by all the threads in the application. The **SessionFactory** interface is available in the **org.hibernate.sessionFactory** and can be obtained by the Configuration object.

There is only one SessionFactory object for an application. If our application is using multiple databases, there will be one SessionFactory object per database. It also holds the second-level cache data in Hibernate.

We can create the SessionFactory object by the following code:

```
Configuration configuration = new Configuration().configure();

SessionFactory factory = configuration.buildSessionFactory();
```

Methods of SessionFactory interface

The SessionFactory interface provides a variety of methods. Some commonlyused methods are listed below:

Methods	Description
public void close()	This method is used to destroy the current SessionFactory and releases all the resources (cache, connection pools). If the SessionFactory is already closed, no-operation will be performed.
public Map getAllClassMetadata()	This method is used to retrieve the ClassMetadata for all mapped entities. It will return a Map containing all ClassMetadata mapped by the corresponding String entityname.
public Map getAllCollectionMetadata()	This method is used to retrieve CollectionMetadata for all mapped collections. It returns a Map from String to CollectionMetadata.
public Cache getCache()	This method can directly access the underlyingcache regions.

public ClassMetadata getClassMetadata(Class entityClass)	This method is used to retrieve the ClassMetadataassociated with the Metadata will become null if there is no entitymapped with it.
public CollectionMetadata getCollectionMetadata(String	This method is used to retrieve

roleName) **public Session getCurrentSession( )**	collection role. If no such Collection is mapped with the Metadata, then it will become null.
	This method is used to obtain the current Hibernate Session and associates the Session with thecurrent Session.
public Statistics getStatistics()	This method is used to retrieve the statistics for thisSessionFactory.
public boolean isClosed()	This method is used to check whether the SessionFactory is open or closed. It will return true if the factory is already closed else false.
public Session openSession()	This method is used to open a Session. The configured connection provider will provide the database connections when needed.
public Session openSession(Connection connection)	This method is used to open a Session using the specified database connection. When we supply a JDBC connection, then the second-level cache will be automatically disabled.
public Session openSession(Interceptor interceptor)	This method is used to open a Session, using the specified interceptor. An interceptor allows user code to intercept or change the property values. The configured connection provider will provide the database connections when needed.
public StatelessSession openStatelessSession()	This method is used to open a new stateless session using the specified database connection. A stateless Session has no persistent object associated with it. It does not implement a first-level cache and a second-level cache.

Hibernate Session Interface

Hibernate Session Interface

A Session is an interface between the Java application and the Hibernate. It is available in the **org.hibernate.session** package. It should not be kept open for a long time, as it is not threadsafe and a short-lived object. A Session instance can be obtained through SessionFactory interface. It is a lightweight object and holds the first-level cache data in Hibernate.

Whenever there is a need for database interaction, a session object is called. The main aim of the Session is to provide communication between the Java application and the database. Session objects are used to execute CRUD (Create, Read, Update, and Delete) operations for the objects of the mapped entity.

Methods of Session Interface

Session interface provides a variety of methods. Some commonly usedmethods are listed below:

Methods	Description
public Transaction beginTransaction()	This method returns the associated transaction object when a unit work is done. If there is no transaction available, then it begins a new transaction.
public void cancelQuery()	This method is used to cancel the execution of the current query. It is the only method of Session which can be safely called from another thread.
public void clear()	This method is used to clear the session completely.
public Connection close()	This method is used to close the connection or end the session. It is not always necessary to close the session.
public Criteria createCriteria(Class persistentClass)	**entityName)**
public Criteria createCriteria(String	

This method is used to create a new Criteria object, for the given persistent class or a	superclass of a persistent class. This method is used to create a new Criteria object, for the given entity name.

public Query createQuery(String queryString)	This method is used to create a new object of Query for the given HQL query string.
public void delete(Object object)	This method is used to remove a persistent object from the database. It cascades to associated objects if the association is mapped with cascade="delete."
public void delete(String entityName, Object object)	It performs the same functions as to **delete(Object object).**
public Serializable getIdentifier(Object object)	This method is used to return the identifier value of the given entity associated with the Session. If the associated object is transient or detached, an exception is thrown.
public Session getSession(EntityMode entityMode)	This method is used to open a new Session with the given entity mode. Here, entityMode is used for creating a new Session. This new Session inherits the Connection, Transaction, and other information from the existing Session.
public SessionFactory getSessionFactory()	This method is used to get the Session factory, which has created the current Session.
public Transaction getTransaction()	This method is used to get the transaction object associated with the current Session.
public void persist(Object object)	This method is used to make a transient object persistent. It cascades to associated objects if the association is mapped with cascade="persist."
public void persist(String entityName, Object object)	It performs the same functions as to **persist(Object object).**
public void setFlushMode(FlushMode flushMode)	This method is used to set the flush mode for the current session. A flush mode defines the points at which the Session is flushed.

public void update(Object object)	This method is used to update the

	persistent object with the identifier of the given detached object. If there exists a persistent object with the same identifier, an exception is thrown.
public void update(String entityName, Object object)	It performs the same functions as to **update(Object object).**

Lifecycle of Hibernate Entities:

The lifecycle of Hibernate entities (Java objects mapped to database tables) involves several states, including:

1. **Transient:** The object is not associated with any Hibernate Session and has no corresponding database record.

2. **Persistent:** The object is associated with a Hibernate Session, and changes to the object are tracked. At this stage, the object may or may not have a corresponding database record.

3. **Detached:** The object was once associated with a Hibernate Session but is no longer. It may have been explicitly detached or due to the session being closed.

4. **Removed:** The object was associated with a Hibernate Session and has been deleted from the database.