What is hibernate?

Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate, and retrieve data from the database.

What are the advantages of Hibernate over JDBC?

Clean Readable Code: Using hibernate, helps in eliminating a lot of JDBC API-based boiler-plate codes

HQL (Hibernate Query Language): Hibernate provides HQL which is closer to Java and is object-oriented in nature. This helps in reducing the burden on developers for writing database-independent queries.

Exception Handling: Hibernate wraps the JDBC exceptions and throws unchecked exceptions like JDBCException or HibernateException.

JDBC vs Hibernate

| JDBC | HIBERNATE |
| --- | --- |
| write code to map the object model’s data representation to the schema | Itself |
| queries and update data to a relational database using SQL | HQL works with object-oriented concepts like inheritance, association |
| try catch block as it throws checked exception | manages the exceptions itself by marking them as unchecked |
| database dependent | database independent |
| Creating associations between relations is not easy | one-to-one, one-to-many, many-to-one, and many-to-many can be acquired easily with the help of annotations. |

What are some of the important interfaces of the Hibernate framework?

Explain hibernate architecture?

Hibernate core interfaces are:

* Configuration
* SessionFactory
* Session
* Criteria
* Query
* Transaction

What is a Session in Hibernate?

A session is an object that maintains the connection between Java object application and database. The session is not thread-safe and the first level cache belongs to the session object. Session also has methods for storing, retrieving, modifying or deleting data from databases using methods like persist(), load(), get(), update(), delete(), etc. Additionally, It has factory methods to return Query, Criteria, and Transaction objects. The Session is represented by an EntityManager.

What is a SessionFactory?

SessionFactory provides an instance of Session. It is a factory class that gives the Session objects based on the configuration parameters in order to establish the connection to the database.

As a good practice, the application generally has a single instance of SessionFactory.

#### What are the states of the object in hibernate?

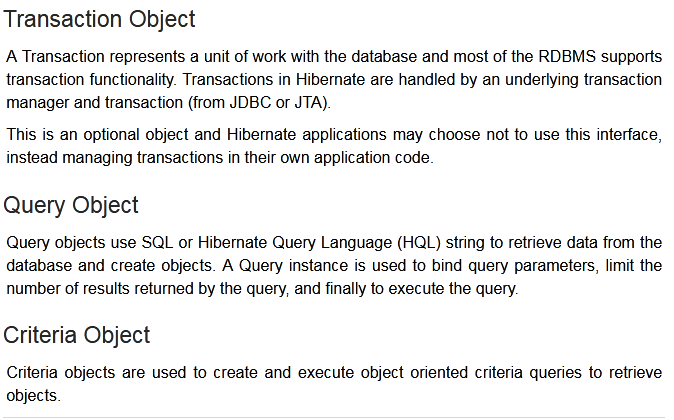
* Instances may exist in one of the following three states at a given point in time −
* transient − A new instance of a persistent class, which is not associated with a Session and has no representation in the database and no identifier value is considered transient by Hibernate.
* persistent − You can make a transient instance persistent by associating it with a Session. A persistent instance has a representation in the database, an identifier value and is associated with a Session.
* detached − Once we close the Hibernate Session, the persistent instance will become a detached instance.

#### Object

#### TranscationObject

#### Query Object

#### Criteria Object



SQL Dialects in Hibernate

* The dialect specifies the type of database used in hibernate so that hibernate generate appropriate type of SQL statements. For connecting any hibernate application with the database, it is required to provide the configuration of SQL dialect.
* Syntax of SQL Dialect
* <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>

why Session Factory is Single for entire DB

* A SessionFactory is very expensive to create, so, for any given database, the application should have only one associated SessionFactory
* The SessionFactory maintains services that Hibernate uses across all Session(s) such as second level caches, connection pools, transaction system integrations, etc

JPA vs. Hibernate

| **JPA** | **Hibernate** |
| --- | --- |
| Java Persistence API (JPA) defines the management of relational data in the Java applications. | Hibernate is an Object-Relational Mapping (ORM) tool which is used to save the state of Java object into the database. |
| It is just a specification. Various ORM tools implement it for data persistence. | It is one of the most frequently used JPA implementation. |
| It is defined in **javax.persistence** package. | It is defined in **org.hibernate** package. |
| The **EntityManagerFactory** interface is used to interact with the entity manager factory for the persistence unit. Thus, it provides an entity manager. | It uses **SessionFactory** interface to create Session instances. |
| It uses **EntityManager** interface to create, read, and delete operations for instances of mapped entity classes. This interface interacts with the persistence context. | It uses **Session** interface to create, read, and delete operations for instances of mapped entity classes. It behaves as a runtime interface between a Java application and Hibernate. |
| It uses **Java Persistence Query Language**(JPQL) as an object-oriented query language to perform database operations. | It uses **Hibernate Query Language** (HQL) as an object-oriented query language to perform database operations. |

What to choose in production

* Validate – will do no changes
* Use tool – Flyway or Liquibase
* Manually applying your database schema is cumbersome and error-prone
* Fortunately, there are technologies for version-controlling your database scripts to automate this process across all environments. In the Java ecosystem, Flyway is one of the most popular and a perfect fit alongside Hibernate and Spring Boot

What is the difference between session.save() and session.persist() method?

| **No.** | **save()** | **persist()** |
| --- | --- | --- |
| 1) | returns the identifier (Serializable) of the instance. | Return nothing because its return type is void. |
| 2) | Syn: public Serializable save(Object o) | Syn: public void persist(Object o) |

What is the difference between get and load method?

The differences between get() and load() methods are given below.

| **No.** | **get()** | **load()** |
| --- | --- | --- |
| 1) | Returns **null** if an object is not found. | Throws **ObjectNotFoundException** if an object is not found. |
| 2) | get() method always **hit the database**. | load() method **doesn't hit** the database. |
| 3) | It returns the real object, not the proxy. | It returns **proxy object.** |
| 4) | It should be used if **you are not sure** about the existence of instance. | It should be used if **you are sure** that instance exists. |

#### 

#### How to do a partial Update?

* @DynamicUpdate

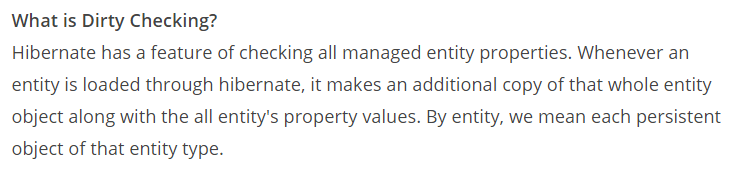
Flyway

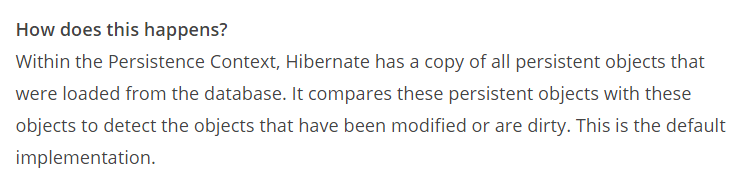
Validate the database schema on application startup with hibernate

**Dirty Checking**

#### What is dirty checking?

#### How does this happen?





Automatic Dirty checking

* Automatic Dirty checking whereby changes to a persistent object are automatically saved to the database when the session is flushed or the transaction is committed. So the code does not need to invoke an explicit save or update

#### GenerationType.IDENTITY

#### lets the database generate a new value with each insert operation.

#### GenerationType

* AUTO: Hibernate selects the generation strategy based on the used dialect,
* IDENTITY: Hibernate relies on an auto-incremented database column to generate the primary key,
* SEQUENCE: Hibernate requests the primary key value from a database sequence,

TABLE: Hibernate uses a database table to simulate a sequence.

* TABLE: Hibernate uses a database table to simulate a sequence.

Hibernate Query Language (HQL)

Hibernate Query Language (HQL) is same as SQL (Structured Query Language) but it doesn't depends on the table of the database. Instead of table name, we use class name in HQL. So it is database independent query language

#### What is the difference between session.save() and session.persist() method?

save(): return type is Java.io.serializable and it returns generated id value.

persist() return type is void.

save():take compare more time to execute

persist():take less time to execute

save():support by hibernate

persist():support by JPA

save(): can be used inside or outside of the transaction

persis(): can be used only inside of the transaction

#### Save vs saveOrUpdate

save() method performs an INSERT operation to store the object into the database, but INSERT will fail if the [primary key](http://java67.blogspot.com/2015/12/difference-between-primary-and-foreign.html) is already persistent, save() used transient object to persistent state.

This is opposite of saveOrUpdate() method, which can do either INSERT or UPDATE SQL query depending upon whether an object exists in the database or not. saveOrUpdate used transient(new) and detached(existing) both into persistent state.

SessionFactory.getCurrentSession()

* And If we talk about SessionFactory.getCurrentSession()
* It creates a new Session if not exists, else uses same session which is in current hibernate context.
* You do not need to flush and close session objects, it will be automatically taken care by Hibernate internally.
* In single threaded environment it is faster than openSession().
* You need to configure additional property. "hibernate.current\_session\_context\_class" to call getCurrentSession() method, otherwise it will throw an exception.

SessionFactory.openSession()

* It always creates a new Session object.
* You need to explicitly flush and close session objects.
* In a single threaded environment it is slower than getCurrentSession().
* You do not need to configure any property to call this method.

#### First Level Cache

Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to the entire application. An application can use many session objects.

#### Second Level Cache

SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to the entire application. But we need to enable it explicitly.

not enabled by default.