Hibernate

**What is Hibernate?**

Hibernate is an open-source Object-Relational Mapping (ORM) framework for Java. It simplifies the development of Java applications that interact with databases by mapping Java objects to database tables and vice versa. This allows developers to work with high-level object-oriented concepts rather than writing complex SQL queries for database interactions. And Hibernate simplifies the data creation, data manipulation and data access. And we can use this for CRUD operations.

**Which API is used by Hibernate?**

Uses the JDBC API to interact with the database

**JPA?**

JPA provides JPQL (Java Persistence Query Language), a query language similar to SQL but operates on the entity objects instead of directly on database tables.

**what are the benefits of hibernate frame work?**

Hibernate is a popular object-relational mapping (ORM) framework for Java. It simplifies the interaction between Java applications and relational databases. Here are some of the key benefits of using Hibernate:

1. **Automatic Table Creation**: Hibernate can automatically create tables based on the entity mappings, reducing the need for manual table creation and synchronization.
2. **Database Independence**: Hibernate abstracts the underlying database, allowing you to switch databases with minimal changes to the code. It supports a variety of databases, including MySQL, PostgreSQL, Oracle, and many others.
3. **HQL (Hibernate Query Language)**: Hibernate provides its own query language, HQL, which is similar to SQL but operates on the entity objects. This allows for more object-oriented querying and abstracts the database-specific SQL dialects.
4. **Caching**: Hibernate supports caching at different levels (first-level and second-level caching). This improves performance by reducing the number of database hits for frequently accessed data.
5. **Lazy Loading**: Hibernate supports lazy loading, which means that related data is only loaded when it is actually accessed. This can significantly improve performance, especially when dealing with large datasets.
6. **Transaction Management**: Hibernate provides built-in transaction management, allowing for consistent and reliable handling of database transactions.
7. **Automatic Dirty Checking**: Hibernate automatically detects changes to persistent entities and updates the database accordingly. This reduces the need for manual updates and ensures data consistency.
8. **Annotation-Based Configuration**: Hibernate supports annotation-based configuration, making it easier to map Java classes to database tables without the need for XML configuration files.
9. **Integration with JPA**: Hibernate implements the JPA (Java Persistence API) specification, allowing developers to use JPA annotations and APIs while benefiting from Hibernate’s advanced features.
10. **Complex Querying and Criteria API**: Hibernate provides a powerful Criteria API for building complex and dynamic

**Annotations?**

* Annotations are placed directly in the Java source code, making it easier to see the configuration while reading the code.
* Reduces the need to manage separate configuration files.

**XML?**

* XML configurations separate the metadata from the source code, promoting a cleaner separation of concerns.
* Easier to manage and change configurations without modifying the Java source code.

**Session Interface:**

* The Session interface is the primary interface for interacting with the database. It represents a single unit of work with the database and provides methods for CRUD operations, transaction management, and query execution.

Key Methods:

* save(), update(), delete(), merge(), get(), load()
* createQuery(), createSQLQuery(), createCriteria()
* beginTransaction(), getTransaction()

**SessionFactory Interface:**

The SessionFactory interface is used to create Session instances. It is a heavyweight object typically created once during application initialization and shared among multiple sessions.

**Key Methods**:

* openSession(), getCurrentSession()
* close()

**Transaction Interface:**

* The Transaction interface provides methods to manage transactions, ensuring that a series of operations are executed in an atomic, consistent, isolated, and durable (ACID) manner.
* **Key Methods**:
* begin(), commit(), rollback()

**Query Interface**:

* The Query interface is used to create and execute HQL (Hibernate Query Language) queries. It provides methods for setting parameters and obtaining query results.
* **Key Methods**:
* setParameter(), setMaxResults(), uniqueResult(), list().

**Criteria Interface:**

* The Criteria interface is used to create and execute queries based on the criteria API, which allows for a programmatic, type-safe way to construct queries.
* Key Methods:

add(), setProjection(), setMaxResults(), list()

**Configuration Interface**:

* The Configuration interface is used to configure Hibernate and build the SessionFactory. It allows for programmatic configuration and addition of annotated classes or XML mappings.
* **Key Methods**:

configure(), addAnnotatedClass(), buildSessionFactory()

**Interceptor Interface**:

* The Interceptor interface provides methods to intercept and manipulate session-related events, such as entity loading, saving, and deleting.
* **Key Methods**:

onLoad(), onSave(), onDelete(), onFlushDirty()

**StatelessSession Interface**:

* The StatelessSession interface is similar to the Session interface but does not maintain a first-level cache or transactional write-behind. It is useful for batch processing and operations that do not require caching or transactions.
* **Key Methods**:

insert(), update(), delete(), createQuery(), beginTransaction()