**Hibernate and JPA Interview Questions and Answers**

**1. What is JPA?**

**Answer:**  
JPA (Java Persistence API) is a **specification** (interface) for object-relational mapping (ORM) in Java. It defines how Java objects are mapped to relational database tables.  
It does **not** provide implementation — frameworks like **Hibernate** provide the actual implementation.

**2. What is Hibernate?**

**Answer:**  
Hibernate is an **ORM tool** and the **most popular JPA implementation**. It handles the persistence of Java objects to database tables and provides additional features like:

* Lazy loading
* Caching
* Automatic schema generation

**3. What is the difference between JPA and Hibernate?**

| **Aspect** | **JPA** | **Hibernate** |
| --- | --- | --- |
| Type | Specification (Interface) | Implementation (Framework) |
| Ownership | Part of Java EE | Open-source ORM by Red Hat |
| Flexibility | Can switch providers | Hibernate-specific features |
| Annotation/API | @Entity, @Id, @OneToMany | Hibernate provides these + more |

**4. What are important annotations in JPA/Hibernate?**

* @Entity: Marks class as a JPA entity.
* @Table: Maps class to a table.
* @Id: Marks primary key.
* @GeneratedValue: Specifies primary key generation strategy.
* @OneToOne, @OneToMany, @ManyToOne, @ManyToMany: Relationship mapping.
* @JoinColumn: Defines foreign key.

**5. What are the types of primary key generation strategies?**

**Answer:**

@GeneratedValue(strategy = GenerationType.AUTO)

Types:

* AUTO
* IDENTITY
* SEQUENCE
* TABLE

**6. What is the difference between get() and load() in Hibernate?**

| **Method** | **Returns Null?** | **Lazy Loading** | **Exception** |
| --- | --- | --- | --- |
| get() | Yes | No | No |
| load() | No (throws exception) | Yes | Yes (if not found) |

**7. What is the first-level cache in Hibernate?**

**Answer:**  
Hibernate maintains a first-level cache (Session cache) by default. Objects fetched in a session are stored in cache — repeated fetch uses cache, not DB.

**8. What is Lazy vs Eager Loading?**

* **Lazy Loading**: Related entities are loaded **on-demand**.
* **Eager Loading**: Related entities are loaded **immediately** with the parent.

@OneToMany(fetch = FetchType.LAZY)

private List<Order> orders;

**9. How do you handle relationships in JPA?**

* @OneToOne
* @OneToMany
* @ManyToOne
* @ManyToMany

📝 Use mappedBy to define bidirectional relationships and cascade for auto-propagation.

**10. What is the use of CascadeType?**

**Answer:**  
Used to propagate persistence operations from parent to child entity.

@OneToMany(cascade = CascadeType.ALL)

private List<Order> orders;

Types include:

* PERSIST
* MERGE
* REMOVE
* REFRESH
* ALL

**11. What is a @NamedQuery?**

@NamedQuery(

name = "Employee.findByName",

query = "SELECT e FROM Employee e WHERE e.name = :name"

)

It is a static, pre-defined query written in JPQL.

**12. What is the difference between merge() and persist()?**

| **Method** | **Behavior** |
| --- | --- |
| persist() | Adds a new entity to the DB (transient → persistent). |
| merge() | Updates detached entity or saves new one if not existing. |

**13. How do you create custom queries in JPA?**

* Use JPQL in @Query

@Query("SELECT c FROM Customer c WHERE c.name = :name")

List<Customer> findByName(@Param("name") String name);

* Use native SQL with nativeQuery = true

**14. How does Hibernate handle transactions?**

Use @Transactional in service layer. It manages commit/rollback automatically.

@Transactional

public void saveCustomer(Customer c) {

repository.save(c);

}

**15. What is EntityManager?**

**Answer:**  
It is the core JPA interface to interact with the persistence context:

* persist()
* merge()
* remove()
* find()
* createQuery()

**16. What is Dirty Checking in Hibernate?**

**Answer:**  
Hibernate tracks entity state. If you modify an object within a session, it automatically updates the DB at transaction commit without needing an explicit update call.

**Advanced & Scenario-Based JPA/Hibernate Questions:**

**1. What is the difference between JPQL and Native SQL?**

* **JPQL**: Uses **entity names and field names**, database independent.
* **Native SQL**: Uses **actual table and column names**, DB-specific features allowed.

**2. How do you handle N+1 select problem in Hibernate?**

* By using:
  + @Fetch(FetchMode.JOIN)
  + JOIN FETCH in JPQL
  + Batch size configuration

@Query("SELECT e FROM Employee e JOIN FETCH e.department")

**3. How does Hibernate manage transactions behind the scenes?**

* Uses **JTA** or **JDBC transactions** based on configuration.
* Transactions are automatically committed or rolled back if used inside @Transactional.

**4. What is the difference between EntityManager.find() and getReference()?**

| **Method** | **Behavior** |
| --- | --- |
| find() | Immediately hits the DB. |
| getReference() | Returns proxy, fetches lazily. |

**5. What is @Embeddable and @Embedded?**

Used to **embed value types** (not entities) inside another entity.

@Embeddable

public class Address {

private String city;

}

@Entity

public class User {

@Embedded

private Address address;

}

**6. What is orphanRemoval in Hibernate?**

When set to true, it deletes child entities if they are removed from the parent collection.

@OneToMany(orphanRemoval = true)

private List<Order> orders;

**7. What are the lifecycle callback annotations in JPA?**

* @PrePersist
* @PostPersist
* @PreUpdate
* @PostUpdate
* @PreRemove
* @PostRemove
* @PostLoad

**8. How can you improve performance in Hibernate?**

* Enable **2nd level cache** or **query cache**.
* Use **batch fetching**.
* Avoid unnecessary joins.
* Use projections or DTOs for read-only queries.

**9. What is optimistic vs pessimistic locking?**

* **Optimistic Locking**: Uses @Version column to detect conflicts.
* **Pessimistic Locking**: Locks rows using SQL-level locks (FOR UPDATE).

**10. How does Hibernate support inheritance?**

Three strategies:

1. **Single Table** – @Inheritance(strategy = InheritanceType.SINGLE\_TABLE)
2. **Table Per Class**
3. **Joined**

**11. What is flush in Hibernate?**

* Synchronizes the persistence context to the database (but doesn’t commit).
* session.flush() forces changes to DB.

**12. What is the difference between Session and EntityManager?**

| **Session (Hibernate)** | **EntityManager (JPA)** |
| --- | --- |
| Hibernate-specific | JPA-standard |
| More methods (e.g., evict()) | Limited to standard API |

**13. What is the persistence context in JPA?**

A set of managed entity instances that exist in a session. Entity changes are tracked automatically.

**14. How can you call a stored procedure in JPA?**

Using @NamedStoredProcedureQuery or EntityManager.createStoredProcedureQuery().

**15. How do you implement soft delete in JPA?**

Add a deleted flag and override queries to exclude soft-deleted rows.