Hibernate Interview Questions

1. What is Hibernate?

Answer: Hibernate is an open-source ORM (Object-Relational Mapping) tool that simplifies the process of handling databases in Java applications. It provides a framework to map an object-oriented domain model to a relational database.

2. What are the benefits of using Hibernate?

Answer: Key benefits include:

- Simplifies database interaction.
- Provides HQL (Hibernate Query Language) to perform database operations.
- Handles complex queries and joins automatically.
- Supports caching for performance improvement.
- It is database independent.

3. What is ORM?

Answer: ORM stands for Object-Relational Mapping. It is a programming technique to map objects in object-oriented programming languages like Java to relational databases.

4. What are the core components of Hibernate?

Answer:

- **Configuration**: Configuration file (hibernate.cfg.xml) for setting up Hibernate.
- **SessionFactory**: Provides session objects for database operations.
- **Session**: Main interface to interact with the database.
- **Transaction**: Manages transaction boundaries.
- **Query**: Executes SQL or HQL queries.

5. What is SessionFactory?

Answer: SessionFactory is a thread-safe object in Hibernate that creates Session objects. It is built at the start of an application and is usually created only once and shared among threads.

6. What is a Session in Hibernate?

Answer: A Session is an interface between Java application and the database, used to perform CRUD operations. It is not thread-safe and should be closed after use.

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

Answer:

- get(): Fetches the object from the database immediately.
- load(): Returns a proxy object and only fetches the data when it is accessed (lazy loading).

8. What is HQL?

Answer: HQL (Hibernate Query Language) is an object-oriented query language similar to SQL but it operates on persistent objects rather than directly on tables.

9. What is the N+1 Select problem in Hibernate?

Answer: It occurs when Hibernate executes N additional queries to retrieve associated entities in a collection, leading to performance issues. It can be avoided using join fetch or the @BatchSize annotation.

10. What is caching in Hibernate?

Answer: Hibernate supports caching mechanisms to improve performance:

- **First-level cache**: Exists at the session level, default cache.
- **Second-level cache**: Exists at the SessionFactory level and can be configured using cache providers like EhCache, Redis.

11. What is lazy loading in Hibernate?

Answer: Lazy loading delays the fetching of associated data until it is accessed, rather than loading it at the time of fetching the parent entity.

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

Answer:

- save(): Returns the generated identifier and saves the object immediately.
- persist(): Does not return the identifier, and the object is saved when the transaction is committed.

13. What is the @Entity annotation?

Answer: The @Entity annotation is used to define a class as a Hibernate entity (a persistent class), meaning it will be mapped to a table in the database.

14. What are the different states of an object in Hibernate?

Answer:

- **Transient**: Object is not associated with a session or database.
- **Persistent**: Object is associated with a session and saved to the database.
- **Detached**: Object was persistent but the session is closed or cleared.

15. What is Criteria in Hibernate?

Answer: Criteria is an API for retrieving entities based on certain conditions. It provides a flexible way to build dynamic queries.

16. What is Cascade in Hibernate?

Answer: Cascade is a mechanism to apply actions (like persist, merge, delete) to the associated entities automatically. It can be controlled using annotations like @Cascade or cascade attributes in @OneToMany.

17. What is FetchType.LAZY and FetchType.EAGER in Hibernate?

Answer: These are fetching strategies:

- FetchType.LAZY: Data is fetched when it is needed (on-demand).
- FetchType.EAGER: Data is fetched immediately, along with the parent entity.

18. What is the difference between Session.save() and Session.saveOrUpdate()?

Answer:

- save(): Inserts the object as a new row in the database.
- saveOrUpdate(): Either saves a new entity or updates the existing entity if it already exists.

19. What is optimistic locking in Hibernate?

Answer: Optimistic locking prevents lost updates by using version control (usually with a version field in the entity). It does not lock the database row but checks the version when the entity is updated.

20. How does Hibernate handle transactions?

Answer: Hibernate manages transactions using the Transaction interface. You can begin, commit, or roll back transactions explicitly within a session.

21. What is @Id in Hibernate?

Answer: @Id annotation is used to mark a field as the primary key of an entity.

22. What is the difference between first-level cache and second-level cache?

Answer:

- **First-level cache**: It is session-level and is enabled by default. Each session has its own cache.
- **Second-level cache**: It is shared across sessions and can be configured using external caching providers.

23. What is a proxy object in Hibernate?

Answer: A proxy object is a placeholder that represents the actual entity in a lazy association. The real entity is only loaded when a method is called on the proxy object.

24. What is the use of @GeneratedValue annotation in Hibernate?

Answer: @GeneratedValue is used to specify how the primary key should be generated (e.g., AUTO, IDENTITY, SEQUENCE, or TABLE).

25. What is the difference between merge() and update()?

Answer:

- update(): Updates the entity if it is in the persistent state.
- merge(): Copies the state of a detached entity into the persistent context.

26. What are Hibernate annotations?

Answer: Hibernate annotations provide metadata configuration directly in Java classes (e.g., @Entity, @Table, @Column, @OneToMany) instead of XML files.

27. What is the use of @ManyToOne annotation in Hibernate?

Answer: @ManyToOne defines a many-to-one relationship between entities. It is used to map the association between entities.

28. What is Session.clear() and when would you use it?

Answer: Session.clear() removes all entities from the first-level cache (session cache) without flushing them to the database. This is useful when you want to reset the session without committing changes.

29. What is Hibernate.initialize()?

Answer: Hibernate.initialize() forces initialization of a proxy object or collection, triggering the loading of data in a lazy-loaded association.

30. How do you map a composite key in Hibernate?

Answer: You can map a composite key in Hibernate using the @Embeddable and @EmbeddedId annotations, where the composite key is represented as an embeddable class.

31. What is flush() in Hibernate?

Answer: flush() synchronizes the session's state with the database but does not commit the transaction. It pushes all pending changes to the database, ensuring consistency between the in-memory state and the database.

32. What is @JoinTable in Hibernate?

Answer: @JoinTable is used to define the intermediate table for many-to-many relationships. It specifies the name of the join table and the join columns from both entities.

33. What is the difference between evict() and clear() in Hibernate?

Answer:

- evict(): Removes a specific object from the session cache.
- clear(): Removes all objects from the session cache.

34. What are transient, persistent, and detached states of an object in Hibernate?

Answer:

- **Transient**: The object is created but not associated with any session.
- **Persistent**: The object is associated with a session and represents a row in the database.
- **Detached**: The object is no longer associated with a session but may still represent a database row.

35. How do you configure second-level caching in Hibernate?

Answer: Second-level caching is configured in the hibernate.cfg.xml file by specifying a cache provider (e.g., EhCache) and enabling caching for specific entities using annotations like @Cacheable and @Cache.

36. What is @Embeddable in Hibernate?

Answer: @Embeddable is used to specify a class whose objects can be embedded in another entity. It is typically used for composite keys or reusable object mappings.

37. How does Hibernate handle composite keys?

Answer: Hibernate handles composite keys using @Embeddable and @EmbeddedId annotations, where the composite key is defined as a separate embeddable class.

38. What is the purpose of the @Version annotation in Hibernate?

Answer: The @Version annotation is used to implement optimistic locking. It ensures that no other transaction can modify an entity while it is being updated by the current transaction.

39. What is the difference between Session.update() and Session.merge()?

Answer:

- update(): Reattaches a detached entity to a session and makes it persistent again.
- merge(): Copies the state of a detached entity onto a persistent entity in the session.

40. What is @Inheritance in Hibernate?

Answer: @Inheritance is used to map inheritance hierarchies in Hibernate. It has three strategies: SINGLE_TABLE, JOINED, and TABLE_PER_CLASS.

41. How does Hibernate support batch processing?

Answer: Hibernate supports batch processing by configuring hibernate.jdbc.batch_size property in the configuration file. This allows executing multiple SQL statements as a batch, reducing database round trips.

42. What is the difference between @ElementCollection and @OneToMany?

Answer:

- @ElementCollection: Used for mapping collections of value types.

- @OneToMany: Used for mapping collections of entity types.

43. What are the different fetching strategies in Hibernate?

Answer: Hibernate provides two fetching strategies:

- **Lazy fetching**: Associated data is fetched on-demand.
- **Eager fetching**: Associated data is fetched immediately.

44. What is Session.refresh() in Hibernate?

Answer: Session.refresh() reloads the object from the database, discarding any in-memory changes made in the session.

45. What is @DiscriminatorColumn used for in Hibernate?

Answer: @DiscriminatorColumn is used in SINGLE_TABLE inheritance mapping to differentiate between different entity types in a single table by storing a discriminator value.

46. What is @OrderColumn in Hibernate?

Answer: @OrderColumn is used to define the order of elements in a collection (e.g., @OneToMany collection) by mapping a column that holds the order index.

47. What is @GeneratedValue(strategy = GenerationType.IDENTITY)?

Answer: It specifies that the primary key value is auto-generated by the database (typically for MySQL, PostgreSQL) using an auto-increment column.

48. What is SessionFactory.close() used for?

Answer: SessionFactory.close() closes the SessionFactory and releases all resources used by it, including database connections and cached data.

49. How do you perform pagination in Hibernate?

Answer: Pagination is achieved using the setFirstResult() and setMaxResults() methods on a query to limit the results returned.

50. What is the purpose of @EntityListeners in Hibernate?

Answer: @EntityListeners allows attaching lifecycle callback methods (e.g., @PrePersist, @PostLoad) to entities to execute custom logic during entity state transitions.

51. What is CascadeType.ALL in Hibernate?

Answer: CascadeType.ALL is a cascading option that applies all possible cascade types (PERSIST, MERGE, REMOVE, REFRESH, DETACH) to associated entities.

52. What is LazyInitializationException?

Answer: It occurs when you try to access a lazy-loaded association outside the context of an open Hibernate session, leading to failure in loading the data.

53. What is @OneToOne mapping in Hibernate?

Answer: @OneToOne is used to define a one-to-one relationship between two entities. It maps a single entity to another entity through a foreign key.

54. What is a Proxy in Hibernate?

Answer: A proxy is a placeholder object in Hibernate that represents an actual entity and helps with lazy loading. It fetches the entity data only when required.

55. What is the role of the @NamedQuery annotation?

Answer: @NamedQuery is used to define static HQL queries that can be referenced by name in the code. It helps with code reusability.

56. How do you implement optimistic locking in Hibernate?

Answer: Optimistic locking can be implemented by using the @Version annotation on an entity, which Hibernate uses to check for concurrent updates.

57. What are the types of inheritance strategies in Hibernate?

Answer:

- **SINGLE_TABLE**: All classes in the hierarchy are stored in a single table.
- **JOINED**: Separate tables for each class, with primary keys joined through foreign keys.
- **TABLE_PER_CLASS**: Separate tables for each class without shared primary keys.

58. What is the use of @MappedSuperclass in Hibernate?

Answer: @MappedSuperclass is used to define common entity mappings that other entities can inherit but is not itself a standalone entity.

59. What is @Temporal used for in Hibernate?

Answer: @Temporal is used to specify the exact type of java.util.Date or java.util.Calendar to be stored in the database (DATE, TIME, TIMESTAMP).

60. How does Hibernate handle transactions?

Answer: Hibernate uses the Transaction interface to demarcate transaction boundaries and can work with both local and global (JTA) transactions.

61. How do you handle many-to-many relationships in Hibernate?

Answer: Many-to-many relationships are handled using @ManyToMany annotation along with a join table (@JoinTable) to map the relationship between two entities.

62. What is the difference between clear() and detach()?

Answer:

- clear(): Removes all objects from the session cache.
- detach(): Removes only the specified object from the session cache.

63. What is @Query annotation used for?

Answer: @Query is used to define custom HQL or SQL queries in the repository layer. It can also be used to create dynamic queries.

64. How do you implement native SQL queries in Hibernate?

Answer: Native SQL queries are implemented using the createSQLQuery() method or by annotating a method with @Query and specifying native SQL.

65. What are the different transaction management strategies in Hibernate?

Answer:

- **JDBC Transactions**: Managed through the JDBC API.
- **JTA Transactions**: Global transactions managed by an application server.
- **Container-Managed Transactions**: Handled by EJB containers.

66. What is Session.lock() in Hibernate?

Answer: Session.lock() is used to reattach a detached object to the session without performing any database operations, using a specific lock mode (e.g., NONE, READ, WRITE).

67. What is the role of @BatchSize in Hibernate?

Answer: @BatchSize is used to improve the performance of lazy loading by loading a batch of associated entities at once, reducing the N+1 Select problem.

68. How does Hibernate manage relationships with @JoinColumn?

Answer: @JoinColumn is used to specify the foreign key column in a relationship (e.g., @OneToOne, @ManyToOne) between entities.

69. What is @ManyToMany mapping in Hibernate?

Answer: @ManyToMany mapping defines a many-to-many association between two entities. It requires an intermediate join table to map the relationship.

70. What is @NaturalId in Hibernate?

Answer: @NaturalId is used to define a unique business key for an entity, which is used in place of the primary key for lookups.

71. What is @Formula in Hibernate?

Answer: @Formula is used to map derived properties in an entity by specifying a custom SQL expression as the formula.

72. What is StatelessSession in Hibernate?

Answer: StatelessSession is a lightweight session interface in Hibernate that does not maintain a first-level cache or persistent context, offering improved performance for bulk operations.

73. What is the use of @SQLDelete in Hibernate?

Answer: @SQLDelete allows you to define a custom SQL DELETE statement for an entity, typically used for implementing soft deletes.

74. What is @Where clause used for in Hibernate?

Answer: @Where is used to add a filtering condition to an entity or collection, allowing only specific rows to be fetched from the database.

75. What is an Interceptor in Hibernate?

Answer: An Interceptor is a callback interface in Hibernate used to intercept and modify entity operations before or after execution, such as save, update, delete, etc.

76. What is a CustomUserType in Hibernate?

Answer: A CustomUserType allows defining a custom mapping between Java objects and SQL types that are not supported by Hibernate's built-in types.

77. What are the different types of caches in Hibernate?

Answer:

- **First-level cache**: Session-scoped and enabled by default.
- **Second-level cache**: SessionFactory-scoped and shared across sessions.

78. What is the @Cache annotation in Hibernate?

Answer: @Cache is used to enable second-level caching for an entity or collection, specifying the cache region and caching strategy (e.g., READ_ONLY, NONSTRICT_READ_WRITE).

79. What is the @NamedNativeQuery annotation in Hibernate?

Answer: @NamedNativeQuery is used to define a named SQL query (native query) that can be reused in the application code.

80. How does Hibernate handle SQL injection?

Answer: Hibernate prevents SQL injection by using parameterized queries (e.g., Query.setParameter()) and criteria queries instead of dynamically constructing SQL strings.

81. What is a PersistentSet in Hibernate?

Answer: A PersistentSet is the Hibernate implementation of a set collection that provides automatic dirty checking, lazy loading, and proxy handling for sets in a Hibernate entity.

82. What are CascadeType.PERSIST and CascadeType.MERGE in Hibernate?

Answer:

- CascadeType.PERSIST: Ensures associated entities are saved when the parent entity is persisted.
- CascadeType.MERGE: Ensures associated entities are updated when the parent entity is merged.

83. What is the role of @TableGenerator in Hibernate?

Answer: @TableGenerator is used to generate unique primary key values using a database table that stores the sequence of IDs.

84. What is the difference between clear() and flush() in Hibernate?

Answer:

- clear(): Removes all entities from the session cache.
- flush(): Synchronizes the session's state with the database without committing the transaction.

85. What is the use of the @Inheritance(strategy = InheritanceType.JOINED) annotation?

Answer: It is used to map the joined table inheritance strategy, where each class in the hierarchy is mapped to its own table, and the tables are joined via foreign keys.

86. How does Hibernate manage database connections?

Answer: Hibernate uses connection pools (e.g., C3P0, HikariCP) to manage database connections efficiently. These pools reuse connections instead of opening a new one for each operation.

87. What is the use of @SequenceGenerator in Hibernate?

Answer: @SequenceGenerator defines a sequence generator that Hibernate uses to generate unique values for primary keys using a database sequence.

88. What is the use of @Any mapping in Hibernate?

Answer: @Any is used to map a polymorphic association to any entity type, storing the entity class and identifier in the same column.

89. What is the role of @OptimisticLocking in Hibernate?

Answer: @OptimisticLocking ensures that concurrent updates do not override each other by checking the version of the entity before committing changes.

90. What is @Filter in Hibernate?

Answer: @Filter is used to apply dynamic filtering conditions to entity queries, allowing you to retrieve only specific records based on runtime criteria.

91. What is ScrollableResults in Hibernate?

Answer: ScrollableResults is used to scroll through query results in chunks, reducing memory usage for large datasets.

92. What are StatelessSession benefits in Hibernate?

Answer: StatelessSession offers improved performance for bulk operations since it does not maintain a first-level cache or dirty checking.

93. What is the use of Session.lock()?

Answer: Session.lock() is used to reattach a detached object to a session without performing any database operations, using a specific lock mode (e.g., NONE, READ, WRITE).

94. What is @Cascade in Hibernate?

Answer: @Cascade is an annotation used to define the cascading behavior of entity associations, such as SAVE, UPDATE, DELETE, etc.

95. What is the purpose of @OrderBy in Hibernate?

Answer: @OrderBy is used to specify the sorting order of elements in a collection (e.g., @OneToMany collection) based on one or more entity fields.

96. What is the use of Session.lock() in Hibernate?

Answer: Session.lock() is used to reattach a detached object to the session without performing any database operations, using a specific lock mode (e.g., NONE, READ, WRITE).

97. How does Hibernate manage relationships with @JoinColumn?

Answer: @JoinColumn is used to specify the foreign key column in a relationship (e.g., @OneToOne, @ManyToOne) between entities.

98. What is @ManyToMany mapping in Hibernate?

Answer: @ManyToMany mapping defines a many-to-many association between two entities. It requires an intermediate join table to map the relationship.

99. What is @NaturalId in Hibernate?

Answer: @NaturalId is used to define a unique business key for an entity, which is used in place of the primary key for lookups.

100. What is the role of @Formula in Hibernate?

Answer: @Formula is used to map derived properties in an entity by specifying a custom SQL expression as the formula.