Hibernate - Practice Sheet

10 Q&A on ORM, caching, N+1, lazy loading, and migrations

Q1. What is Hibernate and how does it map Java objects to database tables?

A: Hibernate is a JPA-compliant ORM. @Entity marks a class as a persistent entity. Field annotations (@Column, @Id) map attributes to columns. At startup Hibernate builds a Metamodel and can auto-generate DDL from the mappings.

Q2. Explain the Hibernate entity lifecycle states.

A: Transient: new object, unknown to Hibernate. Persistent: associated with a Session; changes are tracked. Detached: Session closed, changes not tracked. Removed: scheduled for DELETE. Calling session.save() moves transient to persistent.

Q3. What is the difference between Session.get() and Session.load()?

A: get() hits the database immediately and returns null if not found. load() returns a proxy (lazy placeholder) and defers the SQL hit until a field is accessed. load() throws ObjectNotFoundException at access time if the row is missing.

Q4. What is lazy loading and when does it cause issues?

A: Lazy loading defers the SQL for associations (e.g., @OneToMany) until the collection is accessed. It causes LazyInitializationException when accessed outside an open Session. Fix with JOIN FETCH, @EntityGraph, or keeping the Session open (Open Session in View).

Q5. Describe Hibernate's two levels of caching.

A: L1 (Session cache) is always on; identical IDs within one Session return the same object without a second SQL query. L2 (SessionFactory cache, optional) is shared across Sessions; providers include Ehcache and Redis. @Cacheable enables it per entity.

Q6. How do you map a bidirectional @OneToMany relationship?

A: The @OneToMany side uses mappedBy='fieldName' pointing to the owning side. The @ManyToOne side owns the foreign key column. Always update the owning side to persist changes; the inverse side is for navigation only.

Q7. What is the N+1 select problem and how do you fix it?

A: N+1 occurs when loading N parent entities then running one query per entity to fetch a child collection. Fix with JOIN FETCH in JPQL (loads all in one query), @EntityGraph, or batch-fetching via @BatchSize on the association.

Q8. How does optimistic locking work in Hibernate with @Version?

A: Add a @Version Long field to the entity. Hibernate includes WHERE version=? in every UPDATE. If another transaction modified the row first, the version no longer matches and Hibernate throws OptimisticLockException, preventing a lost update.

Q9. What is HQL and how does it differ from SQL and Criteria API?

A: HQL is object-oriented: queries reference entity names and field names, not table or column names. Hibernate translates HQL to the target SQL dialect. The Criteria API constructs the same queries programmatically in a type-safe manner.

Q10. How do you manage schema migrations with Flyway and Hibernate?

A: Set spring.jpa.hibernate.ddl-auto=validate so Hibernate only checks schema matches entities without changing it. Flyway applies versioned SQL scripts (V1__init.sql, V2__add_column.sql) in order, keeping schema changes explicit and auditable.