JPA - Practice Sheet

10 Q&A on entity lifecycle, JPQL, relationships, and Spring Data

Q1. What is JPA and how does it relate to Hibernate?

A: JPA (Jakarta Persistence API) is a specification defining how Java objects map to relational data. Hibernate is the most popular JPA provider that implements the spec. Code written to the JPA API can switch providers without changes.

Q2. Describe the four JPA entity lifecycle states.

A: New (transient): not associated with a persistence context. Managed: changes tracked, flushed to DB on commit. Detached: context closed, changes ignored. Removed: entity marked for deletion; DELETE runs at flush.

Q3. What does EntityManager.persist(), merge(), and remove() each do?

A: persist() makes a new entity managed (INSERT on flush). merge() copies state of a detached entity onto a managed instance and returns it. remove() marks a managed entity for deletion. None issues SQL immediately that happens at flush/commit.

Q4. How do you write a JPQL query and what makes it portable?

A: JPQL uses entity class names and field names instead of table/column names. Example: SELECT u FROM User u WHERE u.email = :email. The JPA provider translates to the target database dialect, making queries portable across databases.

Q5. What is the difference between @Entity, @MappedSuperclass, and @Embeddable?

A: @Entity maps to its own table. @MappedSuperclass shares mapped fields with subclass entities but has no table of its own. @Embeddable defines a value object whose columns are embedded in the owning entity's table.

Q6. Explain the three JPA inheritance mapping strategies.

A: SINGLE_TABLE: all subclasses in one table, a discriminator column identifies the type fast but nullable columns. JOINED: subclass-specific columns in child tables joined to the parent normalized. TABLE_PER_CLASS: full columns in each table no joins but no

Q7. How do derived query methods work in Spring Data JPA?

A: Spring Data parses the method name at startup and generates JPQL. findByEmailAndActiveTrue() generates SELECT u FROM User u WHERE u.email = ?1 AND u.active = true. Supported keywords include Is, Not, Between, LessThan, OrderBy, etc.

Q8. What is the role of @Transactional and how does propagation work?

A: REQUIRED (default) joins an existing transaction or starts one. REQUIRES_NEW always starts a new transaction, suspending the outer one. MANDATORY throws if there is no active transaction. The proxy intercepts method calls to manage commit/rollback.

Q9. How do you map a composite primary key in JPA?

A: With @EmbeddedId: create a @Embeddable class holding the key fields and annotate the entity field with @EmbeddedId. With @IdClass: annotate the entity with @IdClass(MyKey.class) and repeat the key fields annotated @Id in the entity itself.

Q10. What is a JPQL DTO projection and why use it instead of returning entities?

A: A constructor expression (SELECT new com.app.dto.UserDto(u.id, u.name) FROM User u) returns lightweight DTOs instead of full managed entities. This avoids loading unused columns, prevents accidental lazy-loading, and is ideal for read-only API responses.