Lab7D

Placing the @Transactional annotation on the AccountService class ensures all methods operate within a transactional context, maintaining data consistency and integrity. This allows Spring to manage the session and correctly handle lazy-loaded relationships (Customer and AccountEntry).

Modifying Relationships to LAZY:

Changing all relationships in the Account domain class to LAZY defers the fetching of related entities until they are accessed, improving performance by loading only necessary data.

Removing and Re-adding @Transactional:

Without @Transactional: The application encounters LazyInitializationException errors because the session used for fetching entities is closed before the lazy-loaded properties are accessed.

With @Transactional: The transactional context ensures the session remains open, allowing lazy-loaded properties to be fetched correctly when accessed.

Explanation:

By annotating the AccountService class with @Transactional, the application operates within a transactional context managed by Spring. This keeps the Hibernate session open throughout method execution, enabling the correct fetching of lazy-loaded properties. Therefore, eager loading is no longer necessary, and the application performs efficiently without loading unnecessary data.