Transaction Management and Caching in Spring

Transaction Management

Transaction Management ensures that a series of operations within a transaction are executed in a reliable manner, maintaining data integrity.

What is a Transaction?

A transaction is a sequence of operations performed as a single logical unit of work. A transaction should be atomic, consistent, isolated, and durable (ACID properties).

- **Atomicity:** All operations in a transaction succeed or none of them do.
- **Consistency:** The database remains in a consistent state before and after the transaction.
- **Isolation:** Concurrent transactions do not interfere with each other.
- **Durability:** Once a transaction is committed, it remains so.

Spring's Transaction Management

Spring provides a unified programming model for transaction management, whether using local transactions (e.g., JDBC) or global transactions (e.g., JTA).

- **Declarative Transaction Management**: Uses annotations or XML configuration to define transaction boundaries.
- **Programmatic Transaction Management**: Explicitly manages transactions using the TransactionTemplate or PlatformTransactionManager.

The @Transactional annotation in Spring is used to define transactional boundaries. It helps manage transactions within a Spring application.

Purpose:

- Ensure data consistency and integrity by handling transaction boundaries automatically.
- Support for rollback and commit operations.
- Configurable to define transaction behavior and isolation levels.

- **Transaction Propagation**: Defines how transactions are propagated across method calls (e.g., REQUIRED, REQUIRES_NEW).
- **Isolation Levels**: Determines the visibility of changes made by one transaction to other concurrent transactions (e.g., READ_COMMITTED, SERIALIZABLE).

- **Rollback Rules**: Specifies which exceptions should trigger a rollback.

Caching

Caching improves application performance by storing frequently accessed data in memory, reducing the need to fetch data from slower storage systems.

What is Caching?

Caching involves storing the results of expensive or frequently repeated operations to make future accesses faster.

Spring's Caching Abstraction

Spring provides a caching abstraction that supports various cache providers (e.g., EhCache, Caffeine, Redis) through a consistent API.

Key Annotations:

- **@Cacheable**: Indicates that the result of a method call should be cached.
- **@CachePut**: Updates the cache with the result of a method call.
- **@CacheEvict**: Removes entries from the cache.