Transaction Management and Caching in Spring Boot

Transaction Management in Spring Boot

1. Introduction to Transaction Management
Transaction management is crucial in applications to ensure data integrity and consistency. It follows the ACID properties (Atomicity, Consistency, Isolation, and Durability) to ensure that database operations are completed correctly.

2. Spring Transaction Management
Spring Boot provides comprehensive support for transaction management. There are two primary approaches: declarative and programmatic. Declarative transaction management uses annotations, while programmatic uses APIs.

3. Declarative Transaction Management
Declarative transaction management is achieved using the @Transactional annotation. It allows configuration of transaction propagation behaviors (e.g., REQUIRED, REQUIRES_NEW), isolation levels, rollback rules, and timeouts.

4. Programmatic Transaction Management
Programmatic transaction management provides more control and is done using TransactionTemplate or PlatformTransactionManager. It's used when transactions need to be handled manually in the code.

Caching in Spring Boot

1. Introduction

Caching improves application performance by storing frequently accessed data in memory. There are different types of caching, such as in-memory caching and distributed caching.

2. Spring Caching Abstraction

Spring provides a caching abstraction layer that allows developers to add caching to applications with minimal effort. Key annotations include @Cacheable, @CachePut, @CacheEvict, and @Caching.

3. Cache Providers and Configuration

Spring Boot supports various cache providers like EhCache, Caffeine, and Redis. Configuring a cache manager is necessary to use the caching abstraction effectively.