Panduan Lengkap Thread, Async, dan Concurrency di Java Spring Boot

Bab 1: Pengantar Thread dan Concurrency

Thread adalah unit eksekusi terkecil dalam sebuah proses. Concurrency mengacu pada kemampuan sistem untuk menangani banyak tugas secara bersamaan. Dalam Java, concurrency memungkinkan pengolahan data yang efisien dan responsive, terutama dalam aplikasi berskala besar.

Bab 2: ThreadPool Executor vs @Async Spring

ThreadPoolExecutor adalah API Java standar, sedangkan @Async adalah abstraksi Spring yang memudahkan menjalankan tugas asynchronous tanpa eksplisit membuat thread.
Kelebihan ThreadPool: kontrol lebih besar terhadap thread dan queue.
Kelebihan @Async: lebih mudah dipakai, terintegrasi dengan Spring Lifecycle.

Bab 3: Konfigurasi ThreadPool

Thread pool Spring Boot dapat dikonfigurasi menggunakan AsyncConfigurer:
- corePoolSize: jumlah thread minimum.
- maxPoolSize: jumlah maksimum thread.
- queueCapacity: jumlah maksimum task yang bisa diantre.
- rejectedExecutionHandler: aksi jika antrean penuh.
- threadNamePrefix: prefix nama thread.

Bab 4: Contoh Implementasi Konfigurasi

Contoh konfigurasi thread pool di Spring Boot:

Bab 5: TODO dan AVOID dalam Menggunakan ThreadPool

TODO:
- Tentukan ukuran pool yang sesuai dengan workload.
- Selalu tangani InterruptedException.
- Gunakan logging untuk melacak eksekusi.

AVOID:

- Mengabaikan konfigurasi default tanpa review.
- Membuat thread langsung tanpa executor.
- Memproses blocking call berat di thread async.

Bab 6: Kapan Menggunakan Thread/Async

- Gunakan thread pool ketika pekerjaan bisa dijalankan paralel seperti I/O heavy task.
- Gunakan async saat perlu memecah tugas menjadi non-blocking dan tidak harus segera dikembalikan.
- Hindari async jika pekerjaan harus urut, atau hasilnya penting segera digunakan.