Panduan Pemahaman Thread dan Async di Java Spring Boot

Bab 1: Pengantar Thread dan Konsep Dasar

Thread adalah unit terkecil dari proses yang dapat dijalankan secara konkuren. Dalam Java, thread dapat digunakan untuk melakukan multitasking dalam satu aplikasi. Konsep dasar seperti lifecycle thread, perbedaan antara thread dan proses, serta manfaat thread sangat penting dipahami sebelum masuk ke implementasi.

```
Thread thread = new Thread(() -> {
    System.out.println("Running in thread: " + Thread.currentThread().getName());
});
thread.start();
```

Bab 2: ThreadPoolExecutor dan Konfigurasinya

ThreadPoolExecutor memungkinkan kita mengatur jumlah thread aktif dan antrian task. Ini penting untuk mengontrol resource dan menghindari out-of-memory. Beberapa parameter penting:

- corePoolSize: Jumlah minimal thread yang aktif

- maxPoolSize: Jumlah maksimal thread aktif

- queueCapacity: Jumlah maksimal task yang bisa diantre

- rejectedExecutionHandler: Penanganan saat antrian penuh

```
ExecutorService executor = new ThreadPoolExecutor(
    1, 5, 60L, TimeUnit.SECONDS,
    new ArrayBlockingQueue<>(10),
    new ThreadPoolExecutor.CallerRunsPolicy()
);
```

Bab 3: @Async dan @EnableAsync

Panduan Pemahaman Thread dan Async di Java Spring Boot

@Async digunakan untuk mengeksekusi method secara asynchronous. Agar dapat digunakan, perlu mengaktifkan dengan @EnableAsync. Spring akan secara otomatis menggunakan threadpool default jika tidak dikonfigurasi secara eksplisit.

```
@EnableAsync
@Service
public class MyAsyncService {
    @Async
    public void runAsyncTask() {
        System.out.println("Running async in: " + Thread.currentThread().getName());
    }
}
```

Bab 4: Custom ThreadPool di Spring Boot

Untuk kontrol lebih baik, kita bisa define konfigurasi thread pool sendiri dengan mengimplementasikan AsyncConfigurer dan override getAsyncExecutor().

```
@Configuration
public class AsyncConfig implements AsyncConfigurer {
    @Bean
    public Executor getAsyncExecutor() {
        ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
        executor.setCorePoolSize(5);
        executor.setMaxPoolSize(10);
        executor.setQueueCapacity(25);
        executor.setThreadNamePrefix("MyExecutor-");
        executor.initialize();
        return executor;
    }
}
```

Panduan Pemahaman Thread dan Async di Java Spring Boot

Bab 5: RejectedExecutionHandler dan CallerRunsPolicy

Jika queue penuh dan tidak ada thread tersedia, rejectedExecutionHandler akan menangani task baru. CallerRunsPolicy akan menjalankan task di thread pemanggil, sehingga tidak ada task yang hilang.

```
executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());
```

Bab 6: Studi Kasus: Simulasi 17 Request

Simulasi 17 request dengan core pool 5 dan queue 10. Permintaan ke-16 dan 17 akan menggunakan CallerRunsPolicy. Jika handler lain digunakan, bisa terjadi RejectedExecutionException.

```
IntStream.rangeClosed(1, 17).forEach(i -> {
    executor.submit(() -> {
        Thread.sleep(1000);
    });
});
```