บทที่ 8

Asynchronous Programming

in Spring Boot

หัวข้อที่จะเรียนรู้

- แนวคิด Synchronous vs Asynchronous
- Concurrency ใน Java (Thread, Future, CompletableFuture)
- Spring Boot Async (`@Async`)
- การใช้งานจริง (Long-running tasks, Background jobs)
- การจัดการ Timeout และ Exception

Synchronous vs Asynchronous

Synchronous

- ทำงานเรียงตามลำดับ
- งานหนึ่งเสร็จ → งานถัดไปเริ่ม
- Blocking I/O

Asynchronous

- งานสามารถทำไปพร้อมกันได้
- ไม่ต้องรอผลลัพธ์ก่อนถึงจะทำงานอื่น
- Non-blocking I/O

ตัวอย่าง Synchronous

```
public String generateReport() {
    // ทำงานนาน เช่น ดึงข้อมูลจาก DB
    return "Report finished";
}
```

- Client เรียก → ต้องรอจนเสร็จ

ตัวอย่าง Asynchronous

```
@Async
public CompletableFuture<String> generateReportAsync() {
    return CompletableFuture.supplyAsync(() -> {
        // งานใช้เวลานาน
        return "Report finished";
    });
}
```

- Client เรียก → ได้ Future กลับมาก่อน
- งานยังคงทำงานเบื้องหลัง

เปิดใช้งาน Async ใน Spring Boot

1. เพิ่ม `@EnableAsync` ที่ class หลัก

```
@SpringBootApplication
@EnableAsync
public class MyApplication { ... }
```

2. ใช้ `@Async` ที่ method ที่ต้องการทำงานแบบ async

ตัวอย่าง Service แบบ Async

```
@Service
public class ReportService {
    @Async
    public CompletableFuture<String> generateReport() {
        try {
            Thread.sleep(5000); // จำลองงานที่ใช้เวลานาน
        } catch (InterruptedException e) { }
        return CompletableFuture.completedFuture("Report Done");
    }
}
```

Controller เรียกใช้ Async

```
@RestController
@RequestMapping("/reports")
public class ReportController {
    @Autowired
    private ReportService reportService;

    @GetMapping
    public CompletableFuture<String> getReport() {
        return reportService.generateReport();
    }
}
```

- Client เรียก `/reports` → จะได้ response หลังงานเสร็จ
- ไม่ block thread หลัก

การจัดการ Timeout

- ใช้ `CompletableFuture.orTimeout()`

```
reportService.generateReport()
    .orTimeout(3, TimeUnit.SECONDS)
    .exceptionally(ex -> "Timeout occurred");
```

Lab

1. เปิดใช้งาน `@EnableAsync`
2. เขียน Service ที่ทำงานนาน (simulate 5s) แบบ Async
3. เขียน Controller `/reports` → เรียก service แบบ async
4. ทดสอบเรียกหลายครั้งพร้อมกัน

Assignment

- พัฒนา API `/tasks`
  - **POST** `/tasks` → สร้างงานใหม่ (async)
  - **GET** `/tasks/{id}` → ตรวจสอบสถานะงาน (pending / finished)
  - **GET** `/tasks/{id}/result` → คืนผลลัพธ์เมื่อเสร็จ
- เก็บสถานะงานใน Map/Database
- จัดการ Timeout (งานเกิน 10 วินาทีถือว่าล้มเหลว)