

ISM Lab 04 group 2

Project Overview

The aim of this project was to enhance our understanding of frontend development and aspect-oriented programming (AOP) by creating a practical application. We developed a frontend application using Angular to interact with a backend service for event management, and implemented AOP in Spring Boot to collect and expose usage statistics of the service.

Frontend Development

The frontend was developed using Angular, allowing users to interact with the backend service through a “user-friendly” interface. The key functionalities include viewing and inserting events, and a prototype of registering and logging in.

Event List

Home

Date: 2022-01-01

Location: 123 Main St

Approved: true

Date: 2022-01-01

Location: 123 Main St

Approved: true

Add Event

Home

Email

Password

Repeat Password

Register

Home

Username:

Password:

Login

Aspect-Oriented Programming (AOP)

The AOP implementation was done using Spring Boot to monitor and collect statistics from the EventService. The primary goal was to track the number of times the getAllEvents method was called.

StatisticsAspect (Spring Boot)

- Uses AOP to intercept calls to getAllEvents, increments a counter each time the method is called, and tracks the total number of events retrieved.

```
@Aspect
@Component
public class StatisticsAspect {

    private AtomicInteger getAllEventsCallCount = new AtomicInteger( initialValue: 0);
    private AtomicLong totalEventsRetrievedCount = new AtomicLong( initialValue: 0);

    @Pointcut("execution(* com.example.com_rucinski.service.EventService.getAllEvents(..))")
    public void getAllEventsPointcut() {}

    @AfterReturning(pointcut = "getAllEventsPointcut()", returning = "result")
    public void afterReturningGetAllEvents(List<EventDTO> result) {

        getAllEventsCallCount.incrementAndGet();
        totalEventsRetrievedCount.addAndGet(result.size());
    }

    public int getGetAllEventsCallCount() { return getAllEventsCallCount.get(); }
    public long getTotalEventsRetrievedCount() { return totalEventsRetrievedCount.get(); }
}
```

StatisticsController (Spring Boot)

- Exposes an endpoint to retrieve the statistics collected by the StatisticsAspect.

```
@RestController
public class StatisticsController {

    @Autowired
    private StatisticsAspect statisticsAspect;

    @GetMapping("/api/statistics")
    public ResponseEntity<StatisticsDTO> getStatistics() {
        StatisticsDTO statistics = new StatisticsDTO(
            statisticsAspect.getAllEventsCallCount(),
            statisticsAspect.getTotalEventsRetrievedCount()
        );
        return ResponseEntity.ok(statistics);
    }
}
```

StatisticsDTO

- A simple DTO to structure the statistics data.

```
class StatisticsDTO {
    private int getAllEventsCallCount;
    private long totalEventsRetrievedCount;

    public StatisticsDTO(int getAllEventsCallCount, long totalEventsRetrievedCount) {
        this.getAllEventsCallCount = getAllEventsCallCount;
        this.totalEventsRetrievedCount = totalEventsRetrievedCount;
    }
}
```