테스트 환경의 중요성, 결정성 위배

발표, 비타(오연택)

INDEX

목차

01

상황, 시간 정렬 검증

```java
@Entity
@EntityListeners(...)
public class Announcement {

    @Id
    @GeneratedValue(...)
    private Long id;

    ...

    @CreatedDate
    private LocalDateTime createdAt;
}
```

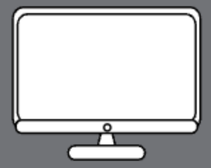
```java
private List<Announcement> sortAnnouncements(
        List<Announcement> announcements,
) {
    return announcements.stream()
            .sorted(createdAtDescending())
            .toList();
}
```

Announcement

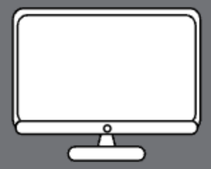
- 공지 사항 Entity 생성 일자 자동 생성 (Auditing)
- 공지 사항의 생성(CreatedAt)일자 기준으로 정렬해 반환

정렬 로직, 검증 테스트

```java
@Test
void 성공_생성일_역순_정렬() {
    // given
    Announcement ..1 = AnnouncementFixture.create(organization);
    Announcement ..2 = AnnouncementFixture.create(organization);
    Announcement ..3 = AnnouncementFixture.create(organization);
    announcementJpaRepository.saveAll(List.of(..1, ..2, ..3));

    // when & then
    List<String> dateTime = RestAssured
            .given()
            .when()
            .get("/announcements")
            .then()
            .extract()
            .jsonPath()
            .getList("pinned.createdAt", String.class);

    List<LocalDateTime> result = dateTime.stream()
            .map(LocalDateTime::parse)
            .toList();

    assertSoftly(s -> {
        s.assertThat(result.get(0)).isEqualTo(..3.getCreatedAt());
        s.assertThat(result.get(1)).isEqualTo(..2.getCreatedAt());
        s.assertThat(result.get(2)).isEqualTo(..1.getCreatedAt());
    });
}
```



Localhost



Localhost


hosted runner

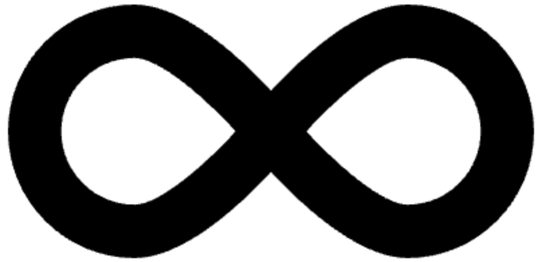
hosted runner

02

문제 원인 탐색

간혈적으로 발생하는 에러인가?

Trouble Shooting



Localhost

hosted runner

로그로 값을 출력해 봐야겠다.

```
List<LocalDateTime> result = dateTime.stream()
        .map(LocalDateTime::parse)
        .toList();

assertSoftly(s -> {
    s.assertThat(result.get(0)).isEqualTo(..3.getCreatedAt());
    s.assertThat(result.get(1)).isEqualTo(..2.getCreatedAt());
    s.assertThat(result.get(2)).isEqualTo(..1.getCreatedAt());
});
```

```
// 출력 == System.out.println
출력("===== createdAt 출력 시작 =====");
for (int i = 0; i < result.size(); i++) {
    출력("[" + i + "] createdAt: " + result.get(i));
}
출력("announcement1: " + ..1.getCreatedAt());
출력("announcement2: " + ..2.getCreatedAt());
출력("announcement3: " + ..3.getCreatedAt());
출력("===== createdAt 출력 끝 =====");
```

```
===== createdAt 출력 시작 =====
pinned[0] createdAt: 2025-07-27T01:03:37.233369
pinned[1] createdAt: 2025-07-27T01:03:37.232973
pinned[2] createdAt: 2025-07-27T01:03:37.228105
announcement1: 2025-07-27T01:03:37.228105
announcement2: 2025-07-27T01:03:37.232973
announcement3: 2025-07-27T01:03:37.233369
===== createdAt 출력 끝 =====
```

IDE JUnit Runner

- 실행 주체: IntelliJ IDEA 자체
- 테스트 엔진: JUnit 5 직접 로드

./gradlew test

```
> Task :processResources
> Task :classes
> Task :compileTestJava
> Task :processTestResources
> Task :testClasses
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath [illegible] appended

> Task :test

AnnouncementControllerTest > getGroupedAnnouncementByOrganizationId > 성공 생성일 역순 정렬 FAILED
    org.assertj.core.error.AssertJMultipleFailuresError at AnnouncementControllerTest.java:203

```

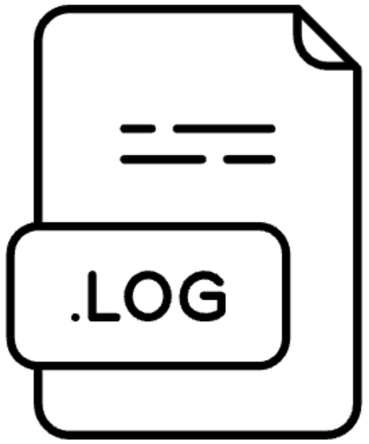
./gradlew test

Gradle Test Runner

- 실행 주체: Gradle 데몬
- 테스트 엔진: Gradle Test Framework

03

테스트 환경 로깅



1 `./gradlew test`

2 

테스트 환경

- **Gradle Test Runner**
  - Gradle이 JVM 환경의 Test task에 따라 실행
  - build.gradle 설정 영향
  - 실패한 테스트 중심의 깔끔하고 빠른 빌드 환경을 제공하려는 설계 철학
- **IDE Test Runner**
  - IntelliJ가 자체적으로 JVM 환경의 JUnit 실행
  - 디버깅/단계별 실행 용이

```groovy
tasks.named('test') {
    useJUnitPlatform()

    testLogging {
        events "failed"
        showStandardStreams = true
    }
}
```

```
./gradlew test --info
```

build.gradle

1. testLogging Events 명시
2. gradlew test의 옵션 명시

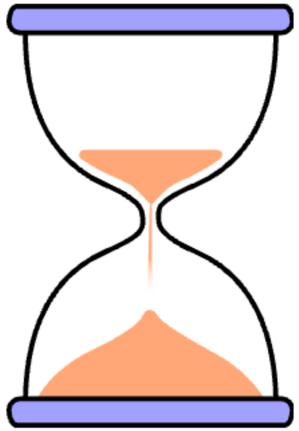
이벤트 이름	설명
`"passed"`	테스트 성공 시 로그 출력
`"failed"`	테스트 실패 시 로그 출력
`"skipped"`	테스트가 건너뛰어진 경우 출력
`"started"`	테스트 시작 시 출력 (잘 안 씀)
`"standard_out"`	`System.out.println()` 출력 포함
`"standard_error"`	`System.err.println()` 출력 포함

```java
// 출력 == System.out.println
출력("===== createdAt 출력 시작 =====");
for (int i = 0; i < result.size(); i++) {
    출력("[" + i + "] createdAt: " + result.get(i));
}
출력("announcement1: " + ..1.getCreatedAt());
출력("announcement2: " + ..2.getCreatedAt());
출력("announcement3: " + ..3.getCreatedAt());
출력("===== createdAt 출력 끝 =====");
```

```
===== createdAt 출력 시작 =====
pinned[0] createdAt: 2025-07-23T12:48:35.709325
pinned[1] createdAt: 2025-07-23T12:48:35.708041
pinned[2] createdAt: 2025-07-23T12:48:35.692878
announcement1: 2025-07-23T12:48:35.692877640
announcement2: 2025-07-23T12:48:35.708040574
announcement3: 2025-07-23T12:48:35.709325377
===== createdAt 출력 끝 =====
```

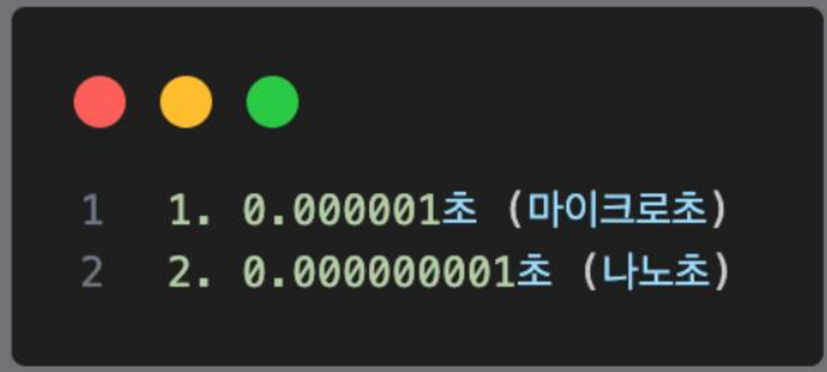
04

OS 환경, 마이크로초와 나노초



```
===== createdAt 출력 시작 =====
pinned[0] createdAt: 2025-07-23T12:48:35.709325
pinned[1] createdAt: 2025-07-23T12:48:35.708041
pinned[2] createdAt: 2025-07-23T12:48:35.692878
announcement1: 2025-07-23T12:48:35.692877640
announcement2: 2025-07-23T12:48:35.708040574
announcement3: 2025-07-23T12:48:35.709325377
===== createdAt 출력 끝 =====
```

```
===== createdAt 출력 시작 =====
pinned[0] createdAt: 2025-07-27T01:03:37.233369
pinned[1] createdAt: 2025-07-27T01:03:37.232973
pinned[2] createdAt: 2025-07-27T01:03:37.228105
announcement1: 2025-07-27T01:03:37.228105
announcement2: 2025-07-27T01:03:37.232973
announcement3: 2025-07-27T01:03:37.233369
===== createdAt 출력 끝 =====
```



Trouble Shooting

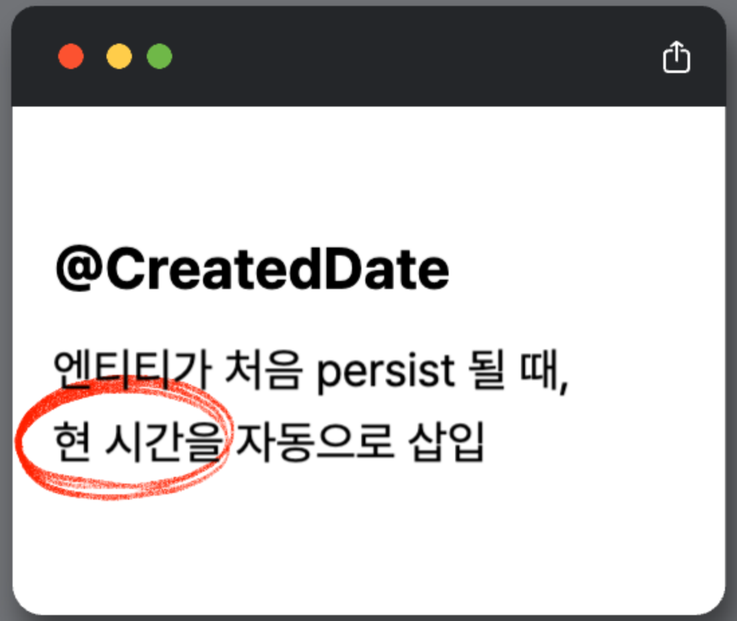
왜, 나노초까지 출력이지?

```java
@CreatedDate
@Column(nullable = false)
private LocalDateTime createdAt;
```

@CreatedDate

엔티티가 처음 persist 될 때,
현 시간을 자동으로 삽입

```java
@CreatedDate
@Column(nullable = false)
private LocalDateTime createdAt;
```



```
1. Clock.systemDefaultZone()
2. System.currentTimeMillis()
```

시간 제공

Mac OS

- 마이크로초 기본 제공

Ubuntu

- 나노초 기본 제공

05

결론, 환경 중요성

환경 차이

시스템에서 시간을 다루는 로직을
LocalDateTime.now() 호출한다고 항상
동일한 결과를 보장 X

또한, 테스트 환경과 실제 환경의 DB 차이가 있을
경우에도 문제 발생 가능성 존재

- DateTime 절삭/반올림
- 인덱스 동작 차이
- 트랜잭션 격리 수준

등 동일한 환경의 중요성

Docker

docker

테스트 환경의 중요성, 결정성 위배