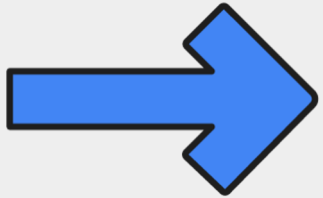
Google Developer Group
Incheon National University / Server

단위 테스트
(Unit Test)

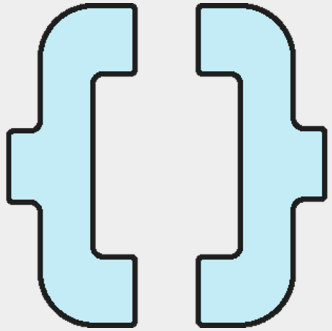


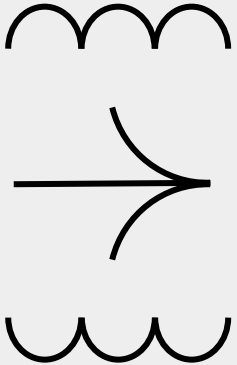
GDGoC INU / SERVER / MEMBER / 이제용

주제

- 단위 테스트란 ?

- 단위 테스트 작성 방법

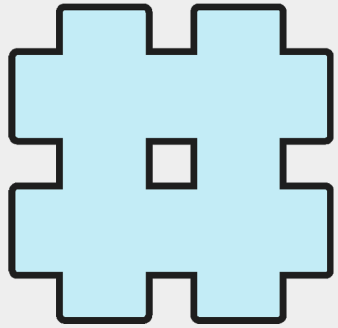




단위 테스트란?

- 소프트웨어의 가장 작은 단위인 모듈, 함수, 클래스, 메서드 등의 개별적인 단위를 분리하여 테스트하는 것

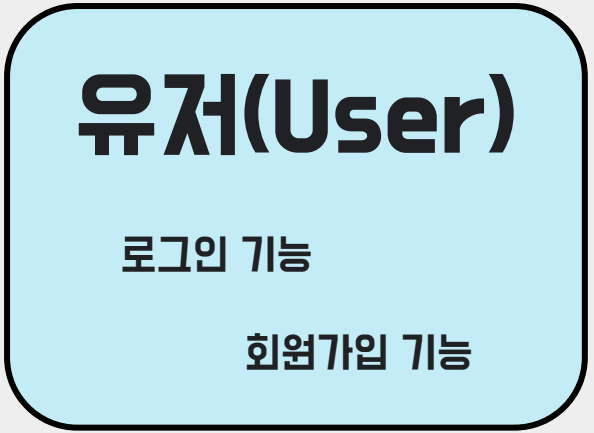
- 개별적인 코드 단위가 설계한 대로 작동하는지 확인하는 행위



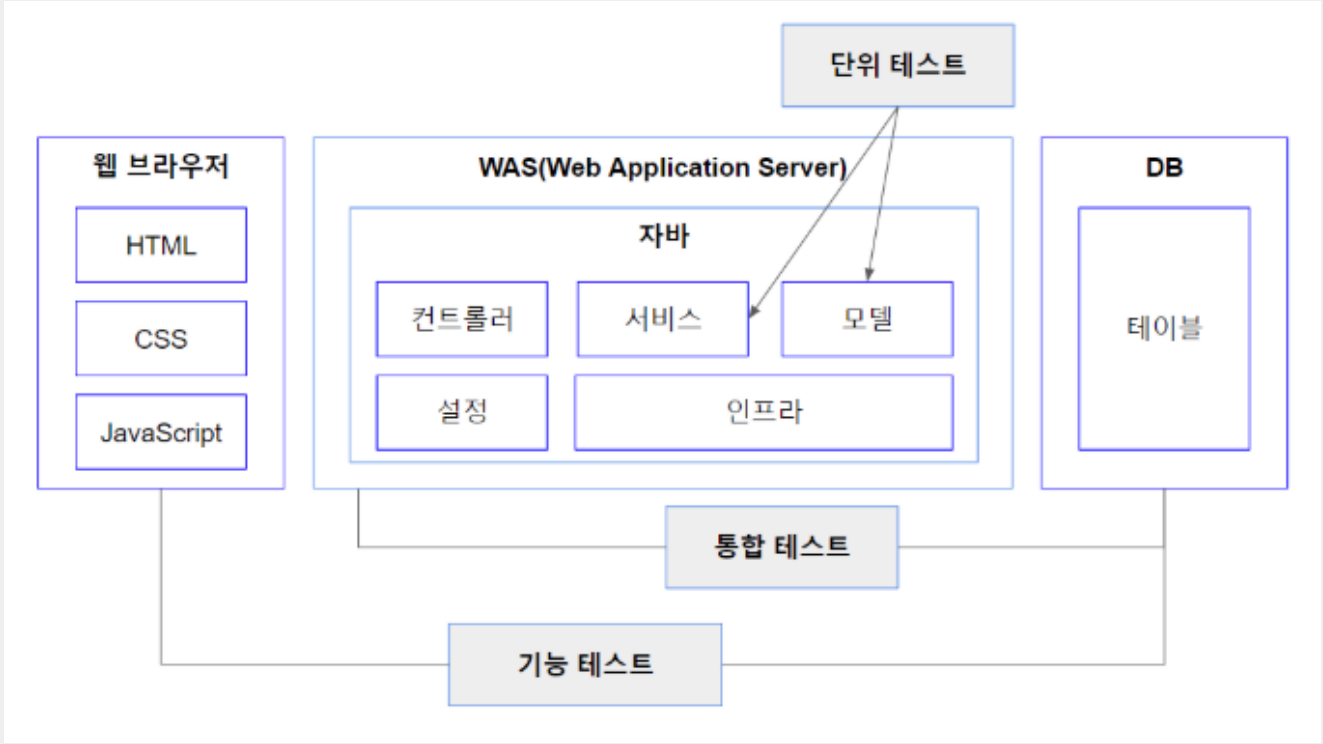
각각의 기능들에 대해 기능 단위로 테스트

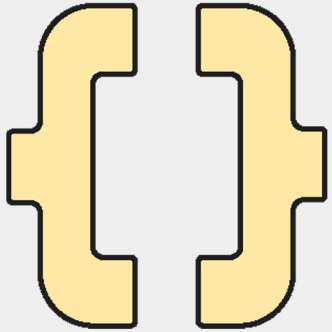
유저의 로그인 기능

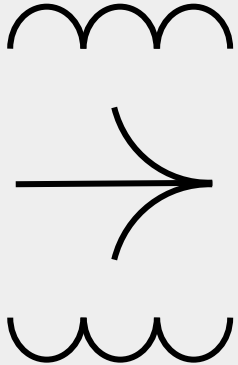
유저의 회원가입 기능



단위테스트의 필요성

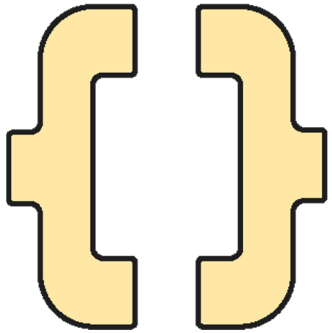

단위 테스트
웹 브라우저
HTML
CSS
JavaScript
WAS(Web Application Server)
자바
컨트롤러
서비스
모델
설정
인프라
DB
테이블
통합 테스트
기능 테스트

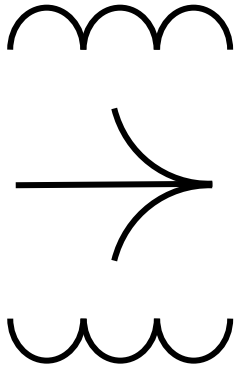




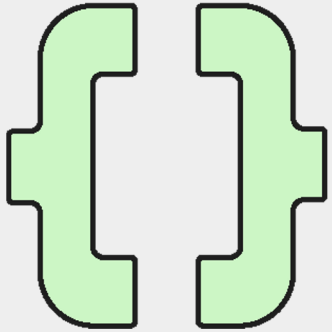
언제 사용하는가?

- 핵심 비지니스 로직 테스트
- 빈번한 코드 변경시
- 개발 초기 단계





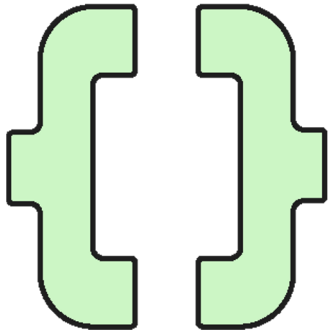
좋은 단위 테스트란?



- 변경되는 요구사항에 맞춰 변경된 코드를 테스트 코드를 통해

  검증함으로써 문제점을 찾아 해결할 수 있어야 한다

좋은 단위 테스트란?

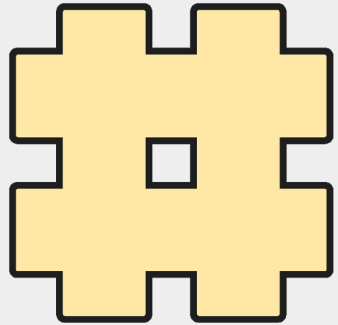


FIRST 규칙 !

- Fast : 테스트는 빠르게 동작하고 자주 가동 해야한다.
- Independent : 각각의 테스트는 독립적이어야 하며, 서로에 대한 의존성은 없어야 한다.
- Repeatable : 어느 환경에서도 반복이 가능해야 한다.
- Self-Validating : 테스트는 성공 또는 실패 값으로 결과를 내어 자체적으로 검증 되어야 한다.
- Timely : 테스트는 테스트 하려는 실제 코드를 구현하기 직전에 구현 해야한다.

테스트 코드 작성을 위한 라이브러리 & 프레임워크

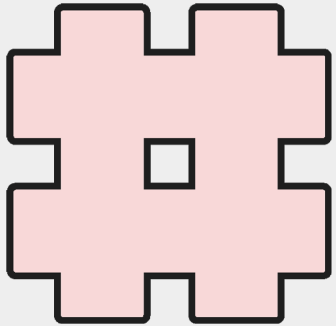
- Junit5
- AssertJ
- Mockito



Given/When/Then 패턴

- Given : 어떠한 데이터가 주어질 때
- When : 어떠한 기능을 실행하면
- Then : 어떠한 결과를 기대한다

```
@Test
@DisplayName("Test")
void test() {
  // Given

  // When

  // Then
}
```



유저 회원가입 기능

유저 로그인 기능

```java
@Service
@Transactional
@RequiredArgsConstructor
public class UserService {

    private final UserRepository userRepository;

    public Long signUp(SignUpRequestDto signUpRequestDto) {

        validateDuplicateMember(signUpRequestDto);
        Users saveUser = buildUser(signUpRequestDto);
        return userRepository.save(saveUser).getId();

    }

    public Long login(LoginRequestDto loginRequestDto) {
        Users user = userRepository.findByEmail(loginRequestDto.getEmail())
                .orElseThrow(() -> new IllegalArgumentException("가입되지 않은 이메일입니다."));

        if (!user.getPassword().equals(loginRequestDto.getPassword())) {
            throw new IllegalArgumentException("비밀번호가 일치하지 않습니다.");
        }

        return user.getId();
    }

}
```

```
@Mock
private UserRepository userRepository;

@InjectMocks
private UserService userService;

private SignUpRequestDto signUpRequestDto;
private LoginRequestDto loginRequestDto;
private Users user;
```

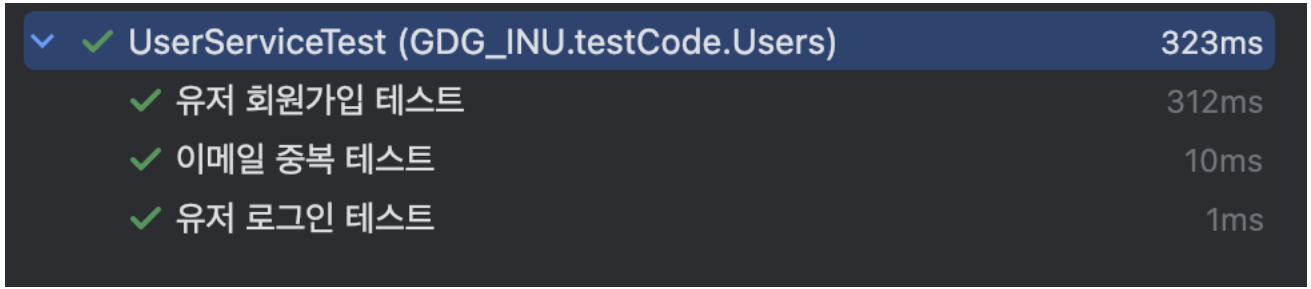
```
@BeforeEach
void setUp() {
    MockitoAnnotations.openMocks(this);

    reset(userRepository);

    signUpRequestDto = SignUpRequestDto.builder()
            .email("test@test.com")
            .password("password")
            .name("testUser")
            .build();

    loginRequestDto = LoginRequestDto.builder()
            .email("test@test.com")
            .password("password")
            .build();

    user = Users.builder()
            .email("test@test.com")
            .password("password")
            .name("testUser")
            .build();
}
```

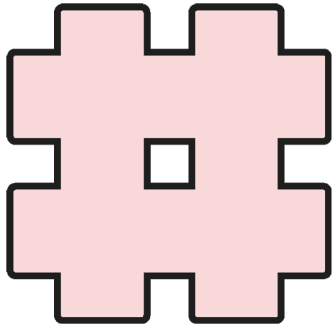
```java
@Test
@DisplayName("유저 회원가입 테스트")
void signUp() {
    // Given
   when(userRepository.existsByEmail(signUpRequestDto.getEmail())
    .thenReturn(false);
    when(userRepository.save(any(Users.class))).thenReturn(user);

    // When
    Long userId = userService.signUp(signUpRequestDto);

    // Then
    Assertions.assertThat(user.getId()).isEqualTo(userId);
    Assertions.assertThat(userRepository.existsByEmail
    (signUpRequestDto.getEmail())).isTrue();
}
```

```java
@Test
@DisplayName("유저 로그인 테스트")
void login() {
    // Given
    when(userRepository.findByEmail(loginRequestDto.getEmail())).
    thenReturn(java.util.Optional.ofNullable(user));

    // When
    Long userId = userService.login(loginRequestDto);

    // Then
    Assertions.assertThat(user.getId()).isEqualTo(userId);
}
```

```java
@Test
@DisplayName("이메일 중복 테스트")
void duplicateEmail() {
    // Given
     when(userRepository.findByEmail(anyString()))
     .thenReturn(Optional.of(user));
    when(userRepository.existsByEmail(anyString()))
     .thenReturn(true);

    // When
    IllegalStateException exception = Assertions.catchThrowableOfType(
            () -> userService.signUp(signUpRequestDto),
            IllegalStateException.class
    );

    // Then
    Assertions.assertThat(exception).isNotNull();
    Assertions.assertThat(exception.getMessage())
     .isEqualTo("이미 존재하는 이메일입니다.");

    verify(userRepository, times(1)).existsByEmail(signUpRequestDto.getEmail());
    verify(userRepository, never()).save(any(Users.class));
}
```

테스트 결과


UserServiceTest (GDG_INU.testCode.Users) 323ms
유저 회원가입 테스트 312ms
이메일 중복 테스트 10ms
유저 로그인 테스트 1ms



Google Developer Group
Incheon National University / Server

Q&A

GDGoC INU / SERVER / MEMBER / 이제용

Google Developer Group
Incheon National University / Server

감사합니다

GDGoC INU / SERVER / MEMBER / 이제용