Spring Boot 2장

2020-04-25 양시영

TDD VS UnitTest

TDD

Test Driven Development

RED -> GREEN -> REFACTORING

Unit Test

기능이 정상작동 하는지 확인하는 테스트 코드

```java
@SpringBootTest
@AutoConfigureMockMvc
public class SecurityTest {

    @Autowired
    MockMvc mockMvc;

    @Test
    @WithAnonymousUser
    public void hello_anonymous() throws Exception {
        mockMvc.perform(get( urlTemplate: "/hello")).andExpect(status().isOk());
    }

    @Test
    @WithAnonymousUser
    public void user_anonymous() throws Exception {
        mockMvc.perform(get( urlTemplate: "/user")).andExpect(status().isUnauthorized());
    }

    @Test
    @WithAnonymousUser
    public void admin_anonymous() throws Exception {
        mockMvc.perform(get( urlTemplate: "/admin")).andExpect(status().isUnauthorized());
    }

    @Test
    @WithMockUser(username = "siyoung", roles = "USER")
    public void user_user() throws Exception {
        mockMvc.perform(get( urlTemplate: "/user")).andExpect(status().isOk());
    }

    @Test
    @WithMockUser(username = "siyoung", roles = "USER")
    public void admin_user() throws Exception {
        mockMvc.perform(get( urlTemplate: "/admin")).andExpect(status().isForbidden());
    }
}
```

Unit Test를 왜 사용할까?

빠른 피드백

눈으로 보지 않아도 검증 가능

사이드 이펙트 판단 가능

그냥…

자신감이 생깁니다.

Lombok

@Getter
@Setter
@*ArgsConstructor
@Builder

@*ArgsConstructor

@Getter
@Setter

```java
public class LegacyUserDTO {

    private Long id;

    private String name;

    private String password;

    private String role;

    public LegacyUserDTO() {}

    public LegacyUserDTO(Long id, String name, String password, String role) {
        this.id = id;
        this.name = name;
        this.password = password;
        this.role = role;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    public User toUser() { return User.builder().id(id).name(name).password(password).role(role).build(); }
}
```

```java
@Getter
@Setter
@NoArgsConstructor
public class UserDTO {
    private Long id;

    @NonNull
    private String name;

    @NonNull
    private String password;

    @NonNull
    private String role;

    @Builder
    public UserDTO(Long id, String name, String password, String role) {
        this.id = id;
        this.name = name;
        this.password = password;
        this.role = role;
    }
    public User toUser() {
        return User.builder().id(id).name(name).password(password).role(role).build();
    }
}
```

단순 반복 작업(노가다) 개선 -> 비즈니스 로직 개발에 집중