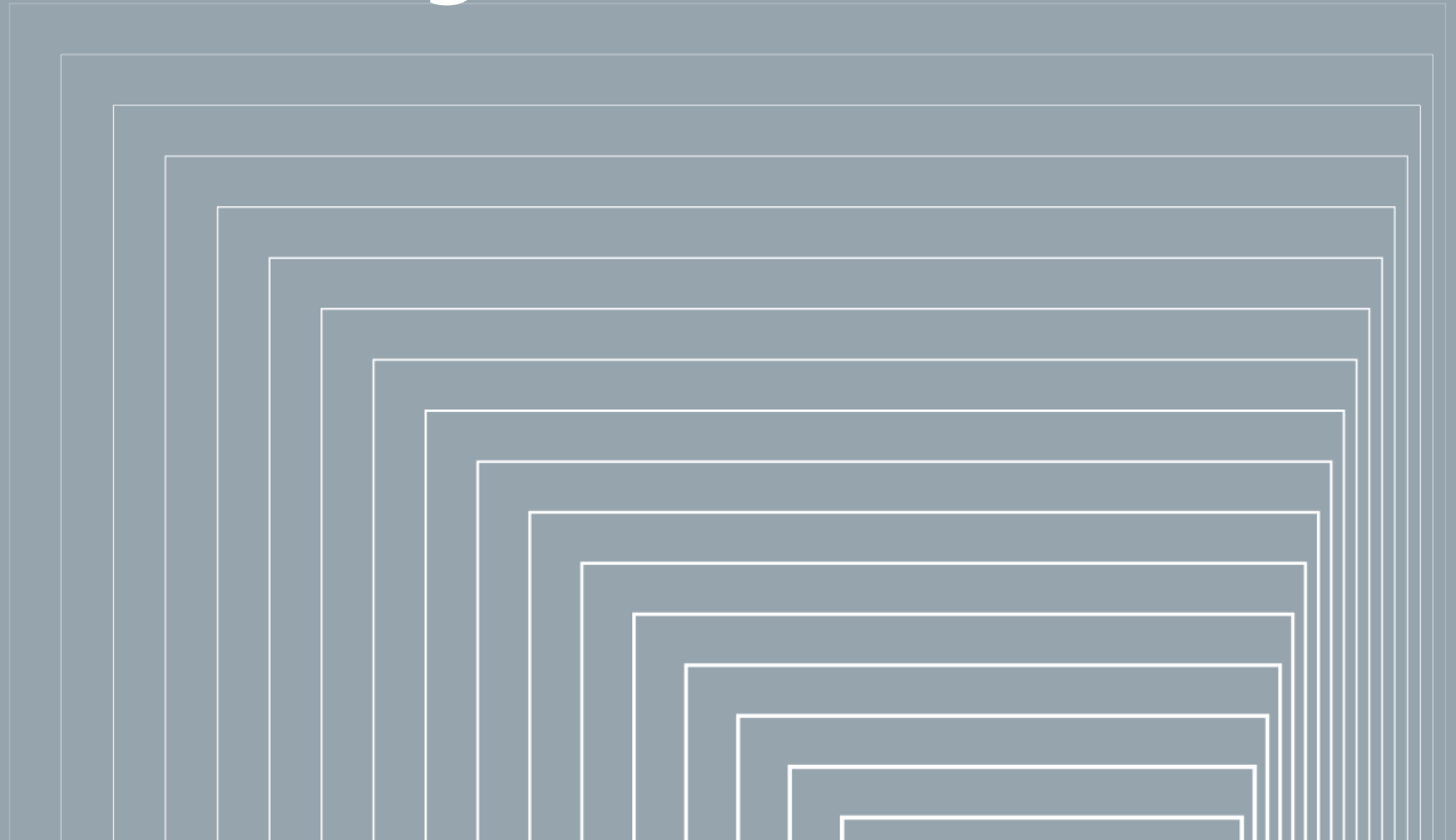
NHN Academy

9. Spring Boot Testing



설치

- spring boot는 테스트를 위한 다양한 애너테이션과 유틸리티를 제공한다.
- 테스트 지원은 spring-boot-test, spring-boot-test-autoconfigure 모듈로 제공된다.
- 개발자는 spring-boot-starter-test 의존성을 추가하여 설치할 수 있다.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

```
dependencies {
   testCompile("org.springframework.boot:spring-boot-starter-test")
}
```

spring-boot-starter-test로 제공하는 라이브러리

구분	설명
JUnit5	Java 애플리케이션 단위 테스트의 산업계 표준(de-facto standard)
Spring Test & Spring Boot Test	Spring Boot 애플리케이션 테스트 지원용 유틸리티와 통합테스트 지원도구
AssertJ	Assertion 라이브러리
Hamcrest	Matcher 객체용 라이브러리
Mockito	Mocking framework
JSONassert	JSON Assertion 용
JsonPath	JSON 의 Xpath 지원

@SpringBootTest

· @SpringBootTest를 사용하면 spring boot의 기능을 사용하면서 통합 테스트 할 때 필요합니다.
· 실제 애플리케이션 기동 시와 거의 유사 하게 전체 빈 설정이 로딩됩니다.

주의

· JUnit 4: @RunWith(SpringRunner.class) 추가
· JUnit 5 : @ExtendWith(SpringExtension.class) 는 이미 포함되어 있음

@SpringBootTest

· webEnvironment를 설정해서 서버를 실행할 수 있다.

SpringBootTest.webEnvironment

구분	설명	비고
MOCK	MockMvc로 테스트 가능	기본
RANDOM_PORT	Embedded WAS 실행, 임의의 포트로 실행, (rollback 동작하지 않음)	@LocalServerPort로 주입
DEFINED_PORT	Embedded WAS 실행, 설정한 포트로 실행, (rollback 동작하지 않음)	server.port 속성으로 결정
NONE	WEB 이 아닌 일반 서비스 테스트용	-

MOCK environment

· MOCK 환경에서는 서버를 실행하지 않기 때문에 MockMvc나 WebTestClient 로 테스트 해야 한다.

```
@SpringBootTest
@AutoConfigureMockMvc
class MyMockMvcTests {
@Autowired
private MockMvc mockMvc;

    @Test
    void testWithMockMvc(@Autowired MockMvc mvc) throws Exception {
        mvc.perform(get("/")).andExpect(status().isOk()).andExpect(content().string("Hello World"));
    }

    // If Spring WebFlux is on the classpath, you can drive MVC tests with a WebTestClient
    @Test
    void testWithWebTestClient(@Autowired WebTestClient webClient) {
        webClient
                .get().uri("/")
                .exchange()
                .expectStatus().isOk()
                .expectBody(String.class).isEqualTo("Hello World");
    }
```

[시연] @SpringBootTest(Mock)

Student 시스템 통합테스트

```
import com.fasterxml.jackson.databind.ObjectMapper;
import com.nhnacademy.edu.springboot.student.Student;
import org.junit.jupiter.api.MethodOrderer;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.TestMethodOrder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.test.web.servlet.MockMvc;

import static org.hamcrest.Matchers.equalTo;
import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*;
import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK)
@AutoConfigureMockMvc
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class StudentControllerTest {
```

[시연] @SpringBootTest(Mock)

Student 시스템 통합테스트 (GET /students)

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK)
@AutoConfigureMockMvc
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class StudentControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    @Order(1)
    void testGetStudents() throws Exception{
        mockMvc.perform(get("/students"))
                .andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[0].name", equalTo("manty")));
    }
```

[시연] @SpringBootTest(Mock)

Student 시스템 통합테스트 (GET /students/{id})

```
@Test
@Order(2)
void testGetStudent() throws Exception{
    mockMvc.perform(get("/students/{id}", 1L))
            .andExpect(status().isOk())
            .andExpect(content().contentType(MediaType.APPLICATION_JSON))
            .andExpect(jsonPath("$.name", equalTo("manty")));
}
```

[시연] @SpringBootTest(Mock)

Student 시스템 통합테스트 (POST /students)

```
@Test
@Order(3)
void testCreateStudent() throws Exception{
    ObjectMapper objectMapper = new ObjectMapper();
    Student zbum = new Student(3L, "zbum1", 100);
    mockMvc.perform(post("/students")
            .content(objectMapper.writeValueAsString(zbum))
                    .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isCreated())
            .andExpect(content().contentType(MediaType.APPLICATION_JSON))
            .andExpect(jsonPath("$.name", equalTo("zbum1")));
}
```

[시연] @SpringBootTest(Mock)

NHN Academy

Student 시스템 통합테스트 (DELETE /students)

```
    @Test
    @Order(4)
    void deleteStudent() throws Exception{
        this.mockMvc.perform(delete("/students/{id}", 3L))
                .andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$.result", equalTo("OK")));
    }
}
```

[실습] Account 시스템 통합테스트

NHN Academy

목표

- Account 시스템의 Controller 메소드를 @SpringBootTest(Mock) 로 통합테스트 작성하세요.

예상시간

- 10분

RANDOM_PORT, DEFINED_PORT

· WebFlux에서 서버를 실행하는 환경에서는 테스트하려면 WebTestClient를 사용 해야 한다.

```
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
class MyRandomPortWebTestClientTests {

    @Test
    void exampleTest(@Autowired WebTestClient webClient) {
        webClient
            .get().uri("/")
            .exchange()
            .expectStatus().isOk()
            .expectBody(String.class).isEqualTo("Hello World");
    }

}
```

RANDOM_PORT, DEFINED_PORT

· WebFlux를 사용할 수 없는 환경에서는 TestRestTemplate을 사용할 수 있다.

```
@SpringBootTest(webEnvironment = WebEnvironment.RANDOM_PORT)
class MyRandomPortTestRestTemplateTests {

    @Test
    void exampleTest(@Autowired TestRestTemplate restTemplate) {
        String body = restTemplate.getForObject("/", String.class);
        assertThat(body).isEqualTo("Hello World");
    }

}
```

[시연] @SpringBootTest(RANDOM_PORT)

NHN Academy

Student 시스템 통합테스트

```
package com.nhnacademy.edu.springboot.student.integration.randomport;

import com.nhnacademy.edu.springboot.student.Student;
import org.junit.jupiter.api.MethodOrderer;
import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.TestMethodOrder;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.*;

import java.util.List;

import static org.assertj.core.api.Assertions.assertThat;

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class StudentControllerTest {
```

[시연] @SpringBootTest(RANDOM_PORT)

Student 시스템 통합테스트 (GET /students)

```
@Autowired
private TestRestTemplate testRestTemplate;

@Test
@Order(1)
void testGetStudents() throws Exception {
    HttpHeaders headers = new HttpHeaders();
    headers.setAccept(List.of(MediaType.APPLICATION_JSON));
    HttpEntity<Student> entity = new HttpEntity<>(headers);
    ResponseEntity<List<Student>> exchange = testRestTemplate.exchange(
            "/students",
            HttpMethod.GET,
            entity,
            new ParameterizedTypeReference<List<Student>>() {
            });

    assertThat(exchange.getBody())
            .contains(new Student(1L, "manty", 100));
}
```

[시연] @SpringBootTest(RANDOM_PORT)

NHN Academy

Student 시스템 통합테스트 (GET /students/{id})

```
@Test
@Order(2)
void testGetStudent() throws Exception{
    ResponseEntity<Student> result = testRestTemplate.getForEntity(
            "/students/{id}",
            Student.class,
            1L);

    assertThat(result.getBody())
            .isEqualTo(new Student(1L, "manty", 100));
}
```

[시연] @SpringBootTest(RANDOM_PORT)

Student 시스템 통합테스트 (POST /students)

```
@Test
@Order(3)
void testCreateStudent() throws Exception{
    Student zbum = new Student(3L, "zbum1", 100);
    ResponseEntity<Student> result = testRestTemplate.postForEntity(
            "/students",
            zbum,
            Student.class);

    assertThat(result.getBody())
            .isEqualTo(zbum);
}
```

[시연] @SpringBootTest(RANDOM_PORT)

NHN Academy

Student 시스템 통합테스트 (DELETE /students)

```
@Test
@Order(4)
void testDeleteStudent() throws Exception{
    testRestTemplate.delete(
            "/students/{id}",
            3L);
}
```

[실습] Account 시스템 통합테스트

NHN Academy

목표

- Account 시스템의 Controller 메소드를 @SpringBootTest(RANDOM_PORT) 로 통합테스트 작성하세요.

예상시간

- 10분

Mocking Beans

- 테스트환경에서 사용할 수 없는 리모트 서비스등을 시뮬레이션 하도록 특정 컴포넌트를 Mocking.
- @MockBean으로 빈을 생성하거나 빈을 대체할 수 있다.

```
@SpringBootTest
class MyTests {

    @Autowired
    private Reverser reverser;

    @MockBean  //RemoteService 를 대체하는 예제
    private RemoteService remoteService;

    @Test
    void exampleTest() {
        given(this.remoteService.getValue()).willReturn("spring");
        String reverse = this.reverser.getReverseValue(); // Calls injected RemoteService
        assertThat(reverse).isEqualTo("gnirps");
    }
```

[시연] @MockBean

Student 시스템 @MockBean 통합테스트

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK)
@AutoConfigureMockMvc
class StudentControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    private StudentRepository studentRepository;

    @Test
    void testGetStudents() throws Exception{
        given(studentRepository.findAll()).willReturn(List.of(new Student(100L, "AA", 90)));

        mockMvc.perform(get("/students"))
                .andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[0].name", equalTo("AA")));
    }
```

[실습] Account 시스템 통합테스트

NHN Academy

목표

· Account 시스템의 Controller 메소드를 @MockBean 을 사용하여 통합테스트를 작성하세요.(4개 메소드)

예상시간

· 10분

Spying Beans

· 테스트환경에서 이미 존재하는 래핑하여 특정 메소드가 다른 동작을 하도록 설정할 수 있다.

```
@SpringBootTest
class MyTests {

    @SpyBean
    private RemoteService remoteService;

    @Autowired
    private Reverser reverser;

    @Test
    public void exampleTest() {
        given(this.remoteService.someCall()).willReturn("mock");
        String reverse = reverser.reverseSomeCall();
        assertThat(reverse).isEqualTo("kcom");
        then(this.remoteService).should(times(1)).someCall();
    }
```

[시연] @SpyBean

Student 시스템 @SpyBean 통합테스트

```
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.MOCK)
@AutoConfigureMockMvc
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class StudentControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @SpyBean
    private StudentService studentService;

    @Test
    @Order(1)
    void testGetStudents() throws Exception {
        given(studentService.getStudents())
                .will(invocation ->{
                    System.out.println("Spy!!");
                    return List.of(new Student(100L, "AA", 90));
                });

        mockMvc.perform(get("/students"))
                .andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[0].name", equalTo("AA")));
    }
```

Auto-configured JSON Tests

· 자동설정 환경에서 객체의 JSON 직렬화, 역직렬화를 테스트하기 위하여 @JsonTest 을 사용합니다.
· AssertJ 기반의 테스트 지원을 제공하기 때문에 객체-JSON 매핑의 결과를 검증할 수 있습니다.

· JacksonTester, GsonTester, JsonbTester, BasicJsonTester 클래스를 각각의 JSON 라이브러리 헬퍼로 사용할 수 있습니다.

Auto-configured JSON Tests

· JacksonTester 사용예

```
@JsonTest
class MyJsonTests {

    @Autowired
    private JacksonTester<VehicleDetails> json;

    @Test
    void serialize() throws Exception {
        VehicleDetails details = new VehicleDetails("Honda", "Civic");
        assertThat(this.json.write(details)).isEqualToJson("expected.json");
        assertThat(this.json.write(details)).hasJsonPathStringValue("@.make");
        assertThat(this.json.write(details)).extractingJsonPathStringValue("@.make")
.isEqualTo("Honda");
    }

    @Test
    void deserialize() throws Exception {
        String content = "{\"make\":\"Ford\",\"model\":\"Focus\"}";
        assertThat(this.json.parse(content)).isEqualTo(new VehicleDetails("Ford", "Focus"));
        assertThat(this.json.parseObject(content).getMake()).isEqualTo("Ford");
```

Auto-configured Spring MVC Tests

· Spring MVC의 Controller를 테스트 하기 위해서는 @WebMvcTest 를 사용한다.
· @WebMvcTest를 사용하면@Controller, @ControllerAdvice, @JsonComponent, Converter, GenericConverter, Filter, HandlerInterceptor, WebMvcConfigurer, WebMvcRegistrations, HandlerMethodArgumentResolver 등 만 스캔한다.
· 테스트에서 다른 컴포넌트를 스캔하고 싶다면 테스트코드에 @Import 로 직접 설정해 주어야 한다.

· Spring Boot Reference Documentation

Auto-configured Spring MVC Tests

· @WebMvcTest 를 사용하면 MockMvc 객체를 얻을 수 있다.

```
@WebMvcTest(UserVehicleController.class)
class MyControllerTests {

    @Autowired
    private MockMvc mvc;

    @MockBean
    private UserVehicleService userVehicleService;

    @Test
    void testExample() throws Exception {
        given(this.userVehicleService.getVehicleDetails("sboot"))
            .willReturn(new VehicleDetails("Honda", "Civic"));
        this.mvc.perform(get("/sboot/vehicle").accept(MediaType.TEXT_PLAIN))
            .andExpect(status().isOk())
            .andExpect(content().string("Honda Civic"));
    }

}
```

Auto-configured Spring MVC Tests

· HtmlUnit 과 Selenium 을 사용한다면 HtmlUnit의 WebClient 도 사용할 수 있다..

```
@WebMvcTest(UserVehicleController.class)
class MyHtmlUnitTests {

    @Autowired
    private WebClient webClient;

    @Test
    void testExample() throws Exception {
        HtmlPage page = this.webClient.getPage("/test");
        assertThat(page.getBody().getTextContent()).isEqualTo("Honda Civic");
    }

}
```

[시연] @SpyBean

Student 시스템 @WebMvcTest

```
@WebMvcTest(SystemController.class)
class SystemControllerTest {

    @Autowired
    MockMvc mockMvc;

    @MockBean
    SystemProperties systemProperties;

    @Test
    void testGetAuthor() throws Exception {
        given(systemProperties.getAuthor())
                .willReturn("ABCDEFG");

        mockMvc.perform(get("/system/author"))
                .andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$.author", equalTo("ABCDEFG")));
    }

}
```

Auto-configured Data JPA Tests

· JPA를 테스트 하기 위해서는 @DataJpaTest 를 사용한다.
· @DataJpaTest 를 사용하면 @Entity, Repository 만 스캔한다.
· 기본적으로 테스트 이후에 수정된 정보는 모두 롤백한다.

· H2와 같은 인메모리 데이터베이스가 클래스 패스에 존재하면 사용하지만 실제 데이터베이스에서 테스트하려면 @AutoConfigureTestDatabase(replace = Replace.NONE) 을 설정해야 한다.

· Spring Boot Reference Documentation

Auto-configured Data JPA Tests

· @DataJpaTest 를 사용하면 TestEntityManager 객체를 얻을 수 있다.

```
@DataJpaTest
class MyRepositoryTests {

    @Autowired
    private TestEntityManager entityManager;

    @Autowired
    private UserRepository repository;

    @Test
    void testExample() throws Exception {
        this.entityManager.persist(new User("sboot", "1234"));
        User user = this.repository.findByUsername("sboot");
        assertThat(user.getUsername()).isEqualTo("sboot");
        assertThat(user.getEmployeeNumber()).isEqualTo("1234");
    }
}
```

[시연] @DataJpaTest

Student 시스템 @DataJpaTest

```
@DataJpaTest
@AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
class StudentRepositorySliceTest {
    @Autowired
    TestEntityManager entityManager;

    @Autowired
    StudentRepository studentRepository;

    @Test
    void testFindAll() {
        Student manty = new Student(10L, "Manty", 100);
        entityManager.merge(manty);

        Student student = studentRepository.findById(10L).orElse(null);
        assertThat(student).isEqualTo(manty);

    }
}
```

[실습] Account 시스템 @DataJpaTest

NHN Academy

목표

· Account 시스템의 Repository 메소드를 @DataJpaTest 을 사용하여 통합테스트를 작성하세요. (findAll, findById, save, delete)

예상시간

· 10분

Test Utilities

- **ConfigDataApplicationContextInitializer**
  - application.properties 를 읽어들이는데 사용한다. @SpringBootTest가 제공하는 모든 기능이 필요 없을때 사용한다.

```
@ContextConfiguration(classes = Config.class, initializers =
ConfigDataApplicationContextInitializer.class)
class MyConfigFileTests {

    // ...

}
```

Test Utilities

- **OutputCapture**
  - System.out, System.err 으로 출력하는 내용을 잡아낼 수 있다..
  - 수정할 수 없는 라이브러리의 결과를 확인할 때 사용할 수 있다.

```
@ExtendWith(OutputCaptureExtension.class)
class MyOutputCaptureTests {

    @Test
    void testName(CapturedOutput output) {
        System.out.println("Hello World!");
        assertThat(output).contains("World");
    }

}
```