

Integration Testing in Spring Boot

Jae Hyeon Kim

— contents

- Integration Testing in Spring Boot
- Example Code / DB

Integration Testing in Spring Boot

— Efficient Solution

- DI (Dependency Injection) 패턴 적용
- SOLID 원칙 적용
- 뷰와 모델의 의존성 분리
- UI 레이어를 분리
- 외부 리소스에 대한 테스트 더미 사용
- 모의 객체 (Mock Object) 활용
- 테스트 자동화
- 지속적인 통합 (CI)

— Dependency Injection

- 코드상 객체를 직접적으로 만드는 게 아닌 밖에서 객체를 넣어주는 방식
- 객체의 의존성 해결 및 관리 목적

```
public class A(){  
    public static void main(String[] args){  
        B b = new B(); //A클래스는 B클래스를 사용한다. 즉, A는 B에 의존한다(의존적이다).  
        b.hello();  
    }  
}  
  
class B(){  
    public void hello(){  
        system.out.print("hello");  
    }  
}
```

— Dependency Injection

- 클래스 모델이나 코드에는 런타임 시점의 의존관계가 드러나지 않는다. 그러기 위해서는 인터페이스만 의존하고 있어야 한다.
- 런타임 시점의 의존관계는 컨테이너나 팩토리 같은 제 3의 존재가 결정한다.
- 의존관계는 사용할 오브젝트에 대한 레퍼런스를 외부에서 제공(주입) 해줌으로써 만들어진다.

– Dependency Injection

```
class BurgerChef {  
    private HamBurgerRecipe hamBurgerRecipe;  
  
    public BurgerChef() {  
        hamBurgerRecipe = new HamBurgerRecipe();  
    }  
}
```

– Dependency Injection 구현 방법(생성자)

```
class BurgerChef {  
    private BurgerRecipe burgerRecipe;  
  
    public BurgerChef(BurgerRecipe burgerRecipe) {  
        this.burgerRecipe = burgerRecipe;  
    }  
}  
  
class BurgerRestaurantOwner {  
    private BurgerChef burgerChef = new BurgerChef(new HamburgerRecipe());  
  
    public void changeMenu() {  
        burgerChef = new BurgerChef(new CheeseBurgerRecipe());  
    }  
}
```


– Dependency Injection 구현 방법(setter)

```
class BurgerChef {  
    private BurgerRecipe burgerRecipe;  
  
    public BurgerChef(BurgerRecipe burgerRecipe) {  
        this.burgerRecipe = burgerRecipe;  
    }  
}  
  
class BurgerRestaurantOwner {  
    private BurgerChef burgerChef = new BurgerChef(new HamburgerRecipe());  
  
    public void changeMenu() {  
        burgerChef = new BurgerChef(new CheeseBurgerRecipe());  
    }  
}
```

— @Autowired

- 의존성 주입을 하려면 IoC 컨테이너에 Bean 등록을 해야함
- Autowired는 필요한 의존 객체의 타입에 해당하는 빈을 찾아 주입한다
- 생성자, setter, 필드 3가지 경우에 autowired를 사용할 수 있다

— @Autowired

```
@Service
public class BugService {
    @Autowired
    private BugRepository bugRepository;
}
```

— How to do Integration Testing

- Springboot에서는 @SpringBootTest를 통해 통합테스트를 진행
- 대부분은 spring-boot-starter-test 의존성을 통해 Spring Boot 테스트 모듈과 Junit Jupiter, AsserJ, Hamcrest 및 기타 여러 유용한 라이브러리를 가져온다.
- 단위 테스트와 같이 기능 검증을 위한 것이 아니라 spring framework에서 전체적으로 플로우가 제대로 동작하는지 검증하기 위해 사용

— Spring-boot-starter-test

- Junit 5 : The de-facto standard for unit testing Java applications
- Spring Test & Spring Boot Test : Utilities and integration test support for spring Boot applications
- AssertJ :A fluent assertion library
- Hamcrest : A library of matcher objects (also known as constraints or predicates)
- Mockito : A Java mocking framework
- JSONassert : An assertion library for JSON
- JsonPath : Xpath for JSON

<https://docs.spring.io/spring-boot/docs/current/reference/html/features.html#features.testing>

— Spring Boot Application Test

- 기본적으로 @SpringBootTest는 서버를 시작하지 않는다. webEnvironment 속성을 사용해 테스트 실행 방법을 더 세분화 할 수 있다.
- MOCK(기본값)
- RANDOM_PORT
- DEFINED_PORT
- NONE

```
@SpringBootTest(  
    properties = {  
        "testId=hiu",  
        "testName=kiku"  
    },  
    webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT  
)
```

WebEnvironment : MOCK

- 실제 객체를 만들기엔 비용과 시간이 많이 들거나 의존성이 길게 걸쳐져 있어 제대로 구현하기 어려운 경우, 가짜 객체를 만들어 사용한다.
- WebApplicationContext를 로드하며 내장된 서블릿 컨테이너가 아닌 Mock 서블릿을 제공한다.
- 별도로 지정하지 않으면 기본값은 Mock 서블릿을 로드하여 구동하게 된다.
- @AutoConfigureMockMvc 어노테이션을 함께 사용하면 별다른 설정 없이 간편하게 MockMvc를 사용한 테스트를 진행할 수 있다.
- MockMvc는 브라우저에서 요청과 응답을 의미하는 객체로서 Controller 테스트를 용이하게 해주는 라이브러리

— WebEnvironment : RANDOM_PORT

- EmbeddedWebApplicationContext를 로드하며 실제 서블릿 환경을 구성 한다.
- 임의의 port listen

— WebEnvironment : DEFINED_PORT

- RANDOM_PORT와 동일하게 실제 서블릿 환경을 구성하지만, 포트는 애플리케이션 프로퍼티에서 지정한 포트를 listen 한다.

— WebEnvironment : NONE

- 기본적인 ApplicationContext를 로드한다.

— Integration Testing : MockMvc

- MockMvc는 웹 어플리케이션을 애플리케이션 서버에 배포하지 않고 테스트용 MVC환경을 만들어 요청 및 전송, 응답기능을 제공하는 유틸리티 클래스
- 모의 요청 및 응답 객체를 생성하여 HTTP 요청을 에뮬레이트
- 컨트롤러 메서드를 직접 호출하고 응답을 확인해 컨트롤러 계층만 테스트

— Integration Testing : MockMvc

1. Static Imports
2. Setup Choices : MockMvcBuilders.webApplicationContextSetup 메서드를 호출해 컨트롤러를 전달하고 MockMvc 인스턴스를 생성
3. Setup Features
4. Performing Requests
5. Defining Expectations

Modifier and Type	Method
static <code>StandaloneMockMvcBuilder</code>	<code>standaloneSetup(Object ... controllers)</code>
static <code>DefaultMockMvcBuilder</code>	<code>webApplicationContextSetup(WebApplicationContext context)</code>

<https://docs.spring.io/spring-framework/docs/current/reference/html/testing.html#spring-mvc-test-framework>

<https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/setup/MockMvcBuilders.html>

— Integration Testing : MockMvc

1. Static Imports
2. Setup Choices : MockMvcBuilders.webApplicationContextSetup 메서드를 호출해 컨트롤러를 전달하고 MockMvc 인스턴스를 생성
3. Setup Features
4. Performing Requests
5. Defining Expectations

Perform 함수 처리 예

request요청처리를 위해 MockMvc의 perform함수를 사용하며 내부에서 MockHttpServletRequest의 값을 설정하여 request요청을 할 수 있다.

```
mockMvc.perform(post("/hotels/{id}", 42).accept(MediaType.APPLICATION_JSON));
```

HTTP메소드 외에 fileUpload메소드를 통해 내부에서 MockMultipartHttpServletRequest의 객체를 만들어 업로드요청을 수행할 수 있다.

```
mockMvc.perform(fileUpload("/doc").file("a1", "ABC".getBytes("UTF-8")));
```

URI template에서 Query String 파라미터를 지정할 수도 있다.

```
mockMvc.perform(get("/hotels?foo={foo}", "bar"));
```

또한 request파라미터를 추가할 수도 있다.

```
mockMvc.perform(get("/hotels").param("foo", "bar"));
```

요청 URI에서 contextPath와 servletPath는 생략하는 것이 바람직하지만 요청시 Full URI와 함께 테스트해야하는 경우, request매핑이 제대로 작동하도록 contextPath와 servletPath를 설정해주도록 한다.

```
mockMvc.perform(get("/app/main/hotels/{id}").contextPath("/app").servletPath("/main"))
```

https://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2:tst:mvc_test

<https://docs.spring.io/spring-framework/docs/current/reference/html/testing.html#spring-mvc-test-framework>

<https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/setup/MockMvcBuilders.html>

— Integration Testing : MockMvc

1. Static Imports
2. Setup Choices : MockMvcBuilders.webApplicationContextSetup 메서드를 호출해 컨트롤러를 전달하고 MockMvc 인스턴스를 생성
3. Setup Features
4. Performing Requests
5. Defining Expectations

response상태 확인 시,

```
mockMvc.perform(get("/accounts/1")).andExpect(status().isOk());
```

andExpect함수를 여러개 사용 가능 시,

```
mockMvc.perform(post("/persons"))  
    .andExpect(status().isOk())  
    .andExpect(model().attributeHasErrors("person"));
```

request요청 결과를 출력할 수도 있다. print메소드를 통해 요청 처리시 관련된 모든 결과 데이터를 출력해준다.

```
mockMvc.perform(post("/persons"))  
    .andDo(print())  
    .andExpect(status().isOk())  
    .andExpect(model().attributeHasErrors("person"));
```

https://www.egovframe.go.kr/wiki/doku.php?id=egovframework:dev2:tst:mvc_test

<https://docs.spring.io/spring-framework/docs/current/reference/html/testing.html#spring-mvc-test-framework>

<https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/test/web/servlet/setup/MockMvcBuilders.html>

– Integration Testing : MockMvc

```
@SpringBootTest(
    properties = {
        "testId=kiku",
        "testName=jaehyun"
    },
    webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT
)
@Transactional
@AutoConfigureMockMvc
@Slf4j
public class MocMVCTesting {
    1개 사용 위치
    @Value("${testId}")
    private String testId;

    1개 사용 위치
    @Value("${testName}")
    private String testName;

    1개 사용 위치
    @Autowired
    MockMvc mvc;
```

```
@Test
void getMember() throws Exception {
    log.info("##### Properties 테스트 #####");
    log.info("testId : " + testId);
    log.info("testName : " + testName);

    log.info("***** START : MOC MVC test *****");
    mvc.perform(get(urlTemplate: "/memberTest/1"))
        .andExpect(status().isOk())
        .andExpect(content().contentType(MediaType.APPLICATION_JSON))
        .andExpect(jsonPath(expression: "$.id", is(value: "kiku")))
        .andDo(print());
    log.info("***** END : MOC MVC test *****");
}
```

— Integration Testing : MockMvc

```
✓ 테스트 통과됨: 1 / 1개 테스트 - 391ms
m1_0,
m1_0.name
from
member m1_0
where
m1_0.mbr_no=?

MockHttpServletRequest:
  HTTP Method = GET
  Request URI = /memberTest/1
  Parameters = {}
  Headers = []
  Body = null
  Session Attrs = {}
```

```
Handler:
  Type = com.example.integrationtesting.controller.TestJpaRestController
  Method = com.example.integrationtesting.controller.TestJpaRestController#getMember(Long)

Async:
  Async started = false
  Async result = null

Resolved Exception:
  Type = null

ModelAndView:
  View name = null
  View = null
  Model = null

FlashMap:
  Attributes = null
```

```
MockHttpServletResponse:
  Status = 200
  Error message = null
  Headers = [Content-Type:"application/json"]
  Content type = application/json
  Body = {"mbrNo":1,"id":"kiku","name":"jaehyun"}
  Forwarded URL = null
  Redirected URL = null
  Cookies = []

2023-03-21T12:56:27.607+09:00 INFO 4872 --- [Test worker] c.e.i.controller.MocMVCTesting : ***** END : MOC MVC test *****
2023-03-21T12:56:27.640+09:00 INFO 4872 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2023-03-21T12:56:27.642+09:00 INFO 4872 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2023-03-21T12:56:27.645+09:00 INFO 4872 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.

BUILD SUCCESSFUL in 6s
4 actionable tasks: 1 executed, 3 up-to-date
오후 12:56:27: 실행이 완료되었습니다 ':test --tests "com.example.integrationtesting.controller.MocMVCTesting.getMember"'.
```


— Integration Testing : TestRestTemplate

- TestRestTemplate은 Spring Boot 애플리케이션에서 RESTful 서비스를 테스트할 수 있도록 Spring Boot에서 제공하는 클래스
- GET, POST, PUT 및 DELETE와 같은 HTTP 요청을 수행하는 여러 메서드를 제공한다
- 요청에 헤더, 쿠키 및 쿼리 매개변수를 추가하는 방법도 제공
- 프로덕션과 유사한 환경에서 실행 중인 애플리케이션에 실제 HTTP 요청을 만드는 통합 테스트를 작성할 수 있다
- MockMvc에 비해 Servlet Container를 사용해 마치 실제 서버가 동작하는 것처럼 테스트를 수행할 수 있다

– Integration Testing : TestRestTemplate

```
@SpringBootTest(
    webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT
)
@Transactional
@AutoConfigureMockMvc
@Slf4j
public class TestRestTemplateTesting {
    1개 사용 위치
    @Autowired
    private TestRestTemplate restTemplate;

    @Test
    void getMember() throws Exception {
        log.info("***** START : TestRestTemplate test *****");
        ResponseEntity<MemberVo> response = restTemplate.getForEntity(url: "/memberTest/1", MemberVo.class);
        then(response.getStatusCode()).isEqualTo(HttpStatus.OK);
        then(response.getBody()).isNotNull();
        log.info(String.valueOf(response.getBody()));
        log.info("***** END : TestRestTemplate test *****");
    }
}
```

Integration Testing : TestRestTemplate

✓ 테스트 통과됨: 1 / 1개 테스트 - 517ms

```
2023-03-22T20:44:09.681+09:00 INFO 20488 --- [    Test worker] c.e.i.c.TestRestTemplateTesting : Started TestRestTemplateTesting in 3.806 seconds (process running f
2023-03-22T20:44:09.890+09:00 INFO 20488 --- [    Test worker] c.e.i.c.TestRestTemplateTesting : ***** START : TestRestTemplate test *****
2023-03-22T20:44:10.025+09:00 INFO 20488 --- [o-auto-1-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-03-22T20:44:10.025+09:00 INFO 20488 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2023-03-22T20:44:10.026+09:00 INFO 20488 --- [o-auto-1-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms
Hibernate:
    select
        m1_0.mbr_no,
        m1_0.id,
        m1_0.name
    from
        member m1_0
    where
        m1_0.mbr_no=?
2023-03-22T20:44:10.194+09:00 INFO 20488 --- [    Test worker] c.e.i.c.TestRestTemplateTesting : MemberVo(mbrNo=1, id=kiku, name=jaehyun)
2023-03-22T20:44:10.194+09:00 INFO 20488 --- [    Test worker] c.e.i.c.TestRestTemplateTesting : ***** END : TestRestTemplate test *****
2023-03-22T20:44:10.227+09:00 INFO 20488 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
2023-03-22T20:44:10.229+09:00 INFO 20488 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
2023-03-22T20:44:10.232+09:00 INFO 20488 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
BUILD SUCCESSFUL in 7s
4 actionable tasks: 2 executed, 2 up-to-date
오후 8:44:10: 실행이 완료되었습니다 ':test --tests "com.example.integrationtesting.controller.TestRestTemplateTesting.getMember"'.
```

Example Code / DB

MemberVO

```
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;

37개 사용 위치
@Data
@AllArgsConstructor
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@Entity(name = "member")
public class MemberVo {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long mbrNo;
    1개 사용 위치
    private String id;
    1개 사용 위치
    private String name;

    @Builder
    public MemberVo(String id, String name) {
        this.id = id;
        this.name = name;
    }
}
```

— TestJpaRestController

```
import com.example.integrationtesting.service.MemberService;
import com.example.integrationtesting.vo.MemberVo;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/memberTest")
public class TestJpaRestController {

    // 기본형
    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    6개 사용 위치
    @Autowired
    MemberService memberService;

    // 모든 회원 조회
    @GetMapping(produces = { MediaType.APPLICATION_JSON_VALUE })
    public ResponseEntity<List<MemberVo>> getAllMembers() {
        List<MemberVo> member = memberService.findAll();
        return new ResponseEntity<List<MemberVo>>(member, HttpStatus.OK);
    }

    // 회원번호로 한명의 회원 조회
    @GetMapping(value = "/{mbrNo}", produces = { MediaType.APPLICATION_JSON_VALUE })
    public ResponseEntity<MemberVo> getMember(@PathVariable("mbrNo") Long mbrNo) {
        Optional<MemberVo> member = memberService.findById(mbrNo);
        return new ResponseEntity<MemberVo>(member.get(), HttpStatus.OK);
    }
}
```

```
// 회원번호로 회원 삭제
@DeleteMapping(value = "/{mbrNo}", produces = { MediaType.APPLICATION_JSON_VALUE })
public ResponseEntity<Void> deleteMember(@PathVariable("mbrNo") Long mbrNo) {
    memberService.deleteById(mbrNo);
    return new ResponseEntity<Void>(HttpStatus.NO_CONTENT);
}

// 회원번호로 회원 수정(mbrNo로 회원을 찾아 Member 객체의 id, name으로 수정함)
@PutMapping(value = "/{mbrNo}", produces = { MediaType.APPLICATION_JSON_VALUE })
public ResponseEntity<MemberVo> updateMember(@PathVariable("mbrNo") Long mbrNo, MemberVo member) {
    memberService.updateById(mbrNo, member);
    return new ResponseEntity<MemberVo>(member, HttpStatus.OK);
}

// 회원 입력
@PostMapping
public ResponseEntity<MemberVo> save(MemberVo member) {
    return new ResponseEntity<MemberVo>(memberService.save(member), HttpStatus.OK);
}

// 회원 입력
@RequestMapping(value = "/saveMember", method = RequestMethod.GET)
public ResponseEntity<MemberVo> save(HttpServletRequest req, MemberVo member){
    return new ResponseEntity<MemberVo>(memberService.save(member), HttpStatus.OK);
}
}
```

— MemberRepository

```
import com.example.integrationtesting.vo.MemberVo;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

4개 사용 위치
@Repository
public interface MemberRepository extends JpaRepository<MemberVo, Long> {
    public List<MemberVo> findById(String id);
    public List<MemberVo> findByName(String name);
    public List<MemberVo> findByNameLike(String keyword);
}
```

MemberService

```
import com.example.integrationtesting.repository.MemberRepository;
import com.example.integrationtesting.vo.MemberVo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

4개 사용 위치
@Service
public class MemberService {
    6개 사용 위치
    @Autowired
    private MemberRepository memberRepository;

    1개 사용 위치
    public List<MemberVo> findAll() {
        List<MemberVo> members = new ArrayList<>();
        memberRepository.findAll().forEach(e -> members.add(e));
        return members;
    }

    2개 사용 위치
    public Optional<MemberVo> findById(Long mbrNo) {
        Optional<MemberVo> member = memberRepository.findById(mbrNo);
        return member;
    }
}
```

```
1개 사용 위치
public void deleteById(Long mbrNo) { memberRepository.deleteById(mbrNo); }

2개 사용 위치
public MemberVo save(MemberVo member) {
    memberRepository.save(member);
    return member;
}

1개 사용 위치
public void updateById(Long mbrNo, MemberVo member) {
    Optional<MemberVo> e = memberRepository.findById(mbrNo);

    if (e.isPresent()) {
        e.get().setMbrNo(member.getMbrNo());
        e.get().setId(member.getId());
        e.get().setName(member.getName());
        memberRepository.save(member);
    }
}
```


Database

이름: test@localhost ☐ DDL 매핑 생성

주석:

일반 옵션 SSH/SSL 스키마 고급

연결 타입: default 드라이버: MariaDB 기타 옵션 ▾

호스트: localhost 포트: 3306

인증: 사용자 및 비밀번호 ▾

사용자: root

비밀번호: <hidden> 저장: 계속 ▾

데이터베이스: test

URL: jdbc:mariadb://localhost:3306/test

위의 설정을 무시합니다.

	MBR_NO	ID	NAME
1	1	kiku	jaehyun
2	2	<null>	<null>
3	3	<null>	<null>
4	4	<null>	<null>
5	5	3333	sdffsd
6	6	sdfsdf	sdfhksjadhfsjdhfk