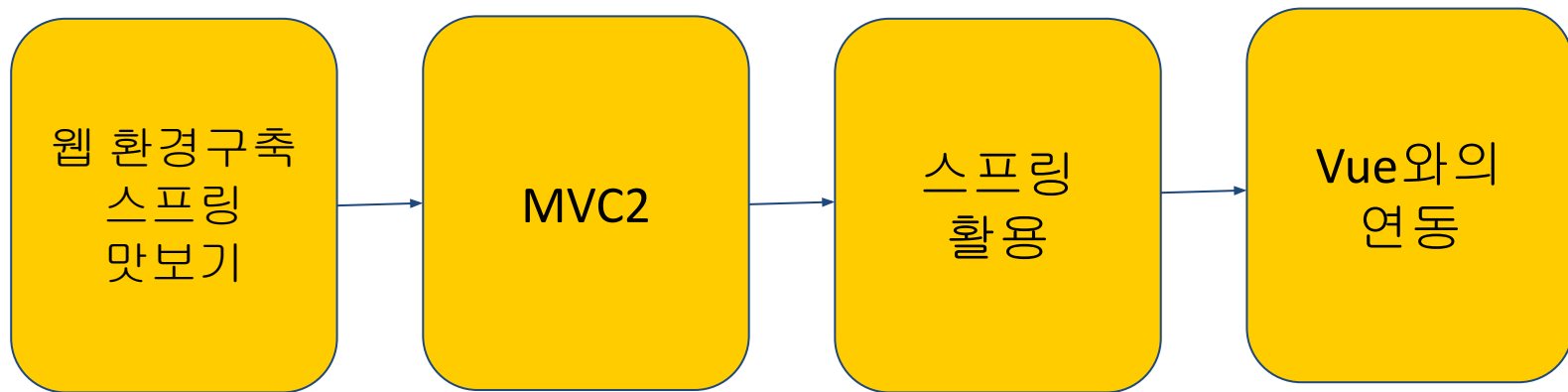


2024년 상반기 K-디지털 트레이닝

SPRING04 - AOP

[KB] IT's Your Life



회원가입 먼저

id:

win55

pw:

....

name:

name

tel:

011

게시판 글쓰기 나중

제목:

title

내용:

content

작성자:

summer1000000000000

글쓰기

```
2024-08-10 12:43:09 - GET "/a3.do", parameters={}
2024-08-10 12:43:09 - Mapped to com.multi.spring2.aop.controller.AOPController#aop3()
anotherTask>> 499500 >> =====
ASiteService.anotherTask() called 1 times, average execution time: 1020ms
performTask>> 499500 >> =====
void com.multi.spring2.aop.service.ASiteService.performTask() executed in 2013ms
anotherTask>> 499500 >> =====
ASiteService.anotherTask() called 2 times, average execution time: 1013ms
performTask>> 499500 >> =====
void com.multi.spring2.aop.service.ASiteService.performTask() executed in 2011ms|
```

AOP

- **관점 지향형 프로그래밍 (AOP: Aspect-Oriented Programming)**
 - AOP는 프로그램에서 핵심 기능과 공통 기능을 분리하여 관리할 수 있게 하는 프로그래밍
 - 특정 페이지에서 구현되어야 하는 핵심 기능과 여러 페이지에서 공통으로 나타나는 기능을 분리
 - 공통적으로 사용되는 기능은 하나의 클래스 파일로 모아두었다가, 실행 시 필요한 곳에 끼워 넣는 방식을 사용

- 공통 관점과 핵심 관점을 나누어 프로그래밍
 - 프로그램을 핵심 기능(핵심 관점)과 공통 기능(공통 관점)으로 나누어 작성
- 공통 관점에 해당하는 기능을 묶어 별도의 클래스로 만든다
 - 공통 기능에 해당하는 부분들을 별도의 클래스에 모아둔다.
- 핵심 관점을 가진 클래스를 만든다
 - 핵심 기능을 담당하는 클래스를 별도로 작성
- 실행 시 핵심 관점 사이에 공통 관점을 끼워 넣는다
 - 실행 시 필요한 곳에 공통 기능을 삽입

A사이트

로그인
개인정보보기
로그아웃

B사이트

물건검색
장바구니
로그인
주문
로그아웃

C사이트(

로그인
주문정보보기
확인
로그아웃

C사이트 : Order interface

COrder.java - 주문정보보기(orderView())
확인(confirm())

빈 이름: order

B사이트 : Product interface

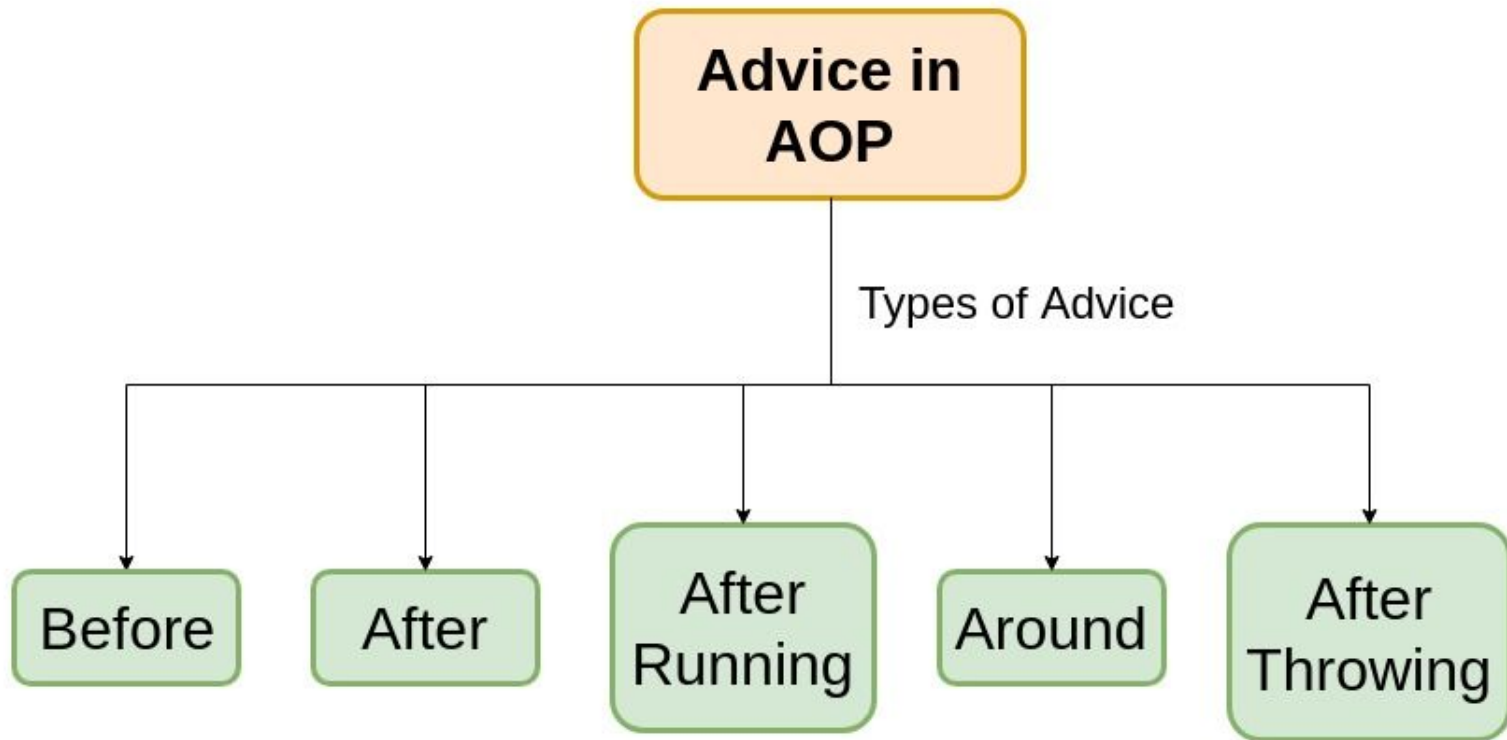
BProduct.java - String 물건검색(search())
장바구니(basket())
주문(orderProduct())

1. 로그인, 로그아웃을 조립!
2. 장바구니에 error=>aop로 에러처리!
3. 물건검색 retrun이 있는 aop메시지 호출!

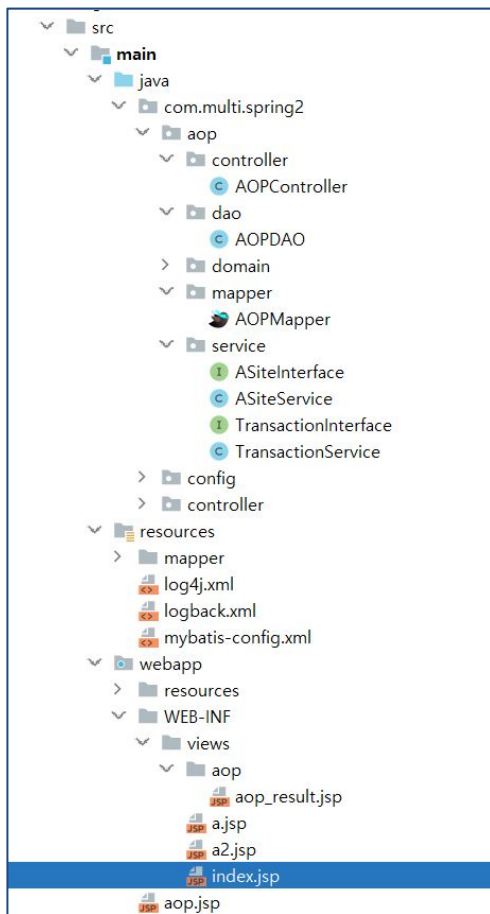
- **OOP vs. AOP**: 객체 지향형 프로그래밍(OOP)과 관점 지향형 프로그래밍(AOP)의 비교
- **OOP**(객체 지향형 프로그래밍):
 - 클래스 기반으로 프로그램을 구성
- **AOP**(관점 지향형 프로그래밍):
 - 메서드 기반으로 프로그램을 구성
 - 핵심적인 관점과 부가적인 관점으로 구분하여 프로그래밍
- **Aspect**(관점):
 - 핵심적인 기능: 그 페이지에 핵심적인 기능
 - 부가적인 기능: 핵심적인 기능 중간에 삽입되는 기능 (횡단 관심사)
- 부가적인 기능을 가진 클래스:
 - **관점 클래스**
 - 예: 로그인/로그아웃

- **프록시 생성**: Spring은 **@Aspect**로 정의된 클래스를 바탕으로 프록시 객체를 생성
 - 이 프록시 객체는 타깃 객체의 메서드를 호출하기 전에 애드바이스를 적용
- **프록시 종류**:
 - **JDK 동적 프록시**: 인터페이스를 기반으로 프록시를 생성
 - a. 타깃 객체가 인터페이스를 구현한 경우 Spring은 기본적으로 JDK 동적 프록시를 사용
 - **CGLIB 프록시**: 타깃 객체가 인터페이스를 구현하지 않은 경우, CGLIB(Code Generation Library)를 사용하여 프록시 객체를 생성
 - a. CGLIB는 타깃 클래스의 서브클래스를 생성하고 메서드를 오버라이드하여 애드바이스를 적용
- **프록시 실행**: 클라이언트가 타깃 객체의 메서드를 호출하면, 사실상 프록시 객체의 메서드가 호출
 - 프록시 객체는 요청을 가로채고, 정의된 애드바이스(예: **@Before**, **@After**, **@Around**)를 실행한 후, 실제 타깃 객체의 메서드를 호출
- **애드바이스 적용**: 프록시는 메서드 호출 전후에 애드바이스를 적용하거나, 메서드 실행을 제어
 - 이를 통해 애플리케이션의 핵심 로직에 영향을 주지 않고 횡단 관심사를 쉽게 추가할 수

- 트랜잭션 관리
- 보안 검사
- 성능 모니터링 등 다양한 기능을 **AOP**를 통해 구현



- Advice in AOP
 - AOP에서의 'Advice'는 특정 메소드가 호출될 때 실행되는 부가적인 행동을 정의하는 것을 의미
- **Before**: 대상 메소드가 실행되기 전에 실행되는 Advice
 - 예를 들어, 메소드 호출 전에 로깅을 하거나 유효성 검사를 할 때 사용
- **After**: 대상 메소드가 정상적으로 실행된 후에 실행되는 Advice
 - 메소드가 완료된 후 리소스를 정리하거나 로깅을 할 때 유용
- **After Running**: 메소드가 실행된 후, 정상적으로 종료되었을 때 실행되는 Advice
 - 메소드의 결과를 바탕으로 추가적인 작업을 수행할 때 사용
- **Around**: 메소드 실행 전과 후에 모두 실행되는 Advice
 - 메소드 실행 자체를 제어할 수 있으며, 메소드 실행 전후에 필요한 로직을 삽입할 수 있음.
- **After Throwing**: 메소드 실행 중 예외가 발생했을 때 실행되는 Advice
 - 예외 처리 로직을 통합 관리하기 위해 사용



```
<body>
<h1></h1>
<br/>
<table>
<tbody>
<tr>
<td style="background: lemonchiffon; text-align: center">
<a href="aop.jsp">
<button>aop.jsp</button>
</a>
</td>

<td style="background: lemonchiffon; text-align: center">
<a href="a.do">
<button>a.do</button>
</a>
</td>

<td style="background: lemonchiffon; text-align: center">
<a href="a2.do">
<button>a2.do</button>
</a>
</td>
</tr>
</tbody>
</table>
</body>
```

```
// 스프링
```

```
implementation 'org.springframework:spring-core:5.3.37'  
implementation "org.springframework:spring-context:5.3.37"  
implementation "org.springframework:spring-webmvc:5.3.37"  
implementation 'javax.inject:javax.inject:1'  
implementation 'org.springframework:spring-web:5.3.37'  
implementation 'org.springframework:spring-jdbc:5.3.37'  
implementation 'com.fasterxml.jackson.core:jackson-databind:2.15.2' // Jackson 라이브러리 추가  
implementation 'com.googlecode.json-simple:json-simple:1.1.1'
```

```
// AOP
```

```
implementation 'org.springframework:spring-context-support:5.3.37'  
implementation 'org.aspectj:aspectjrt:1.9.20'  
implementation 'org.aspectj:aspectjweaver:1.9.20'  
implementation 'org.springframework:spring-tx:5.3.37'
```

```
// JSP, SERVLET, JSTL
```

```
implementation('javax.servlet:javax.servlet-api:4.0.1')  
compileOnly 'javax.servlet.jsp:jsp-api:2.1'  
implementation 'javax.servlet:jstl:1.2'
```

```
@Configuration
@EnableAspectJAutoProxy(proxyTargetClass = true)
public class AOPConfig {

    public AOPConfig() {
        System.out.println("AppConfig created");
    }

    @Bean
    public AspectClass2 aspect() {
        return new AspectClass2();
    }

    @Aspect
    public static class AspectClass2 {

        @Pointcut("execution(* com.multi.spring2.aop.controller.AOPController.aop1*(..))")
        public void tour() {
            // Pointcut method, no implementation needed
        }

        @Before("tour()")
        public void Login() {
            System.out.println("Login aspect triggered");
        }

        @After("tour()")
        public void Logout() {
            System.out.println("Logout aspect triggered");
        }
    }
}}
```

```
@Configuration
@EnableAspectJAutoProxy(proxyTargetClass = true)
public class AOPConfig2 {

    public AOPConfig2() {
        System.out.println("AppConfig2 created");
    }

    @Bean
    public AspectClass2 aspect() {
        return new AspectClass2();
    }

    @Aspect
    public static class AspectClass2 {

        @Pointcut("execution(* com.multi.spring2.aop.controller.AOPController.aop2*(..))")
        public void order() {
        }

        @Before("order()")
        public void Login() {
            System.out.println("Login aspect triggered");
        }

        @After("order()")
        public void Logout() {
            System.out.println("Logout aspect triggered");
        }
    }
}
```



```
public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    public WebAppInitializer() {
        System.out.println("WebAppInitializer created");
    }

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] {
            AppConfig.class,
            AOPConfig.class,
            AOPConfig2.class
        };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { WebConfig.class };
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

    @Override
    protected Filter[] getServletFilters() {
        CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();
        characterEncodingFilter.setEncoding("UTF-8");
        characterEncodingFilter.setForceEncoding(true);
        return new Filter[] { characterEncodingFilter };
    }
}
```

```
@Controller
public class AOPController {

    ASiteInterface aSiteService;
    TransactionInterface aopService;

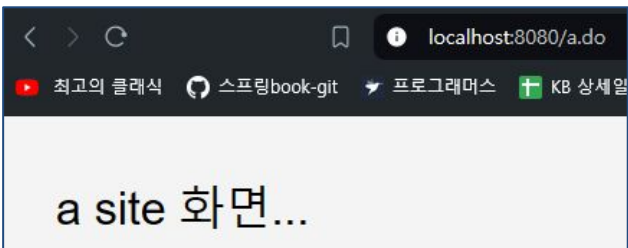
    @Autowired
    public AOPController(ASiteInterface aSiteService, TransactionService aopService) {
        this.aSiteService = aSiteService;
        this.aopService = aopService;
    }

    @RequestMapping("a.do")
    public void aop1() {
        aSiteService.tour();
    }
}
```

```
public interface ASiteInterface {  
    public void tour();  
}
```

```
@Component  
public class ASiteService implements  
ASiteInterface {  
    public void tour() {  
        System.out.println("view personal  
info");  
    }  
}
```

```
<%@ page contentType="text/html; charset=UTF-8"  
language="java" %>  
<html>  
<head>  
    <title>Title</title>  
    <link rel="stylesheet"  
href="../resources/css/out2.css">  
</head>  
<body>  
a site 화면...  
</body>  
</html>
```

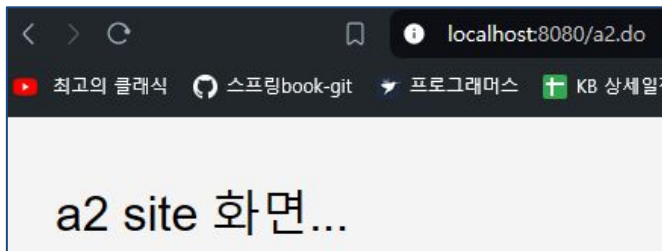


```
Login aspect triggered  
view personal info  
Logout aspect triggered
```

```
public interface ASiteInterface {  
    public void tour();  
    public void order();  
}
```

```
@Component  
public class ASiteService implements ASiteInterface {  
    public void tour() {  
        System.out.println("view personal info  
        -----");  
    }  
  
    @Override  
    public void order() {  
        System.out.println("order product ----- ");  
    }  
}
```

```
<%@ page contentType="text/html; charset=UTF-8"  
    language="java" %>  
<html>  
    <head>  
        <title>Title</title>  
        <link rel="stylesheet"  
            href="../resources/css/out2.css">  
    </head>  
    <body>  
        a2 site 화면...  
    </body>  
</html>
```



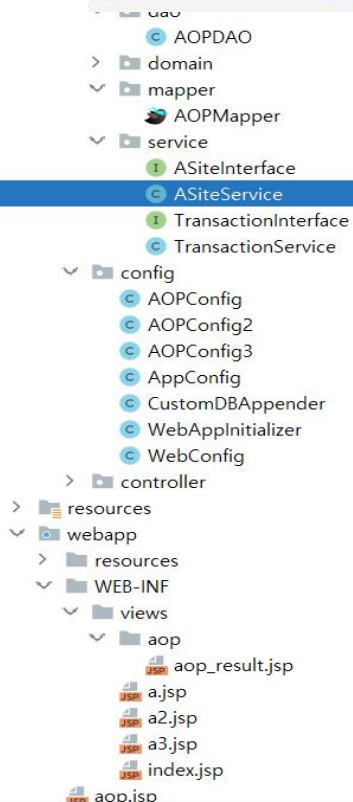
```
Login aspect triggered  
order product -----  
Logout aspect triggered
```

aop.jsp

a.do(ASite)

a2.do(BSite)

a3.do(Performance)



```
<tr>
  <td style="background: lemonchiffon; text-align: center">
    <a href="aop.jsp">
      <button>aop.jsp</button>
    </a>
  </td>

  <td style="background: lemonchiffon; text-align: center">
    <a href="a.do">
      <button>a.do(ASite)</button>
    </a>
  </td>

  <td style="background: lemonchiffon; text-align: center">
    <a href="a2.do">
      <button>a2.do(BSite)</button>
    </a>
  </td>

  <td style="background: lemonchiffon; text-align: center">
    <a href="a3.do">
      <button>a3.do(Performance)</button>
    </a>
  </td>
</tr>
</tbody>
</table>
```

```
@Configuration
@EnableAspectJAutoProxy (proxyTargetClass = true)
public class AOPConfig3 {
```

```
    public AOPConfig3() {
        System.out.println("AppConfig3 created");
    }
```

```
    @Bean
    public AOPConfig3.AspectClass3 aspect() {
        return new AOPConfig3.AspectClass3();
    }
```

```
    @Aspect
    public static class AspectClass3 {
```

```
        private Map<String, Long> methodExecutionCount = new HashMap<>();
        private Map<String, Long> methodExecutionTime = new HashMap<>();
```

```
        @Around("execution(* com.multi.spring2.aop.service.ASiteService.ano*(..))")
        public Object recordExecutionStats(ProceedingJoinPoint joinPoint) throws Throwable {
            String methodName = joinPoint.getSignature().toShortString();
```

```
            long start = System.currentTimeMillis();
            Object proceed = joinPoint.proceed();
            long executionTime = System.currentTimeMillis() - start;
```

```
            methodExecutionCount.put(methodName, methodExecutionCount.getOrDefault(methodName, 0L) + 1);
            methodExecutionTime.put(methodName, methodExecutionTime.getOrDefault(methodName, 0L) + executionTime);
```

```
            long count = methodExecutionCount.get(methodName);
            long totalTime = methodExecutionTime.get(methodName);
            long averageTime = totalTime / count;
```

```
            System.out.println(methodName + " called " + count + " times, average execution time: " + averageTime + "ms");
```

```
            return proceed;
        }
```

```
        @Around("execution(*
com.multi.spring2.aop.service.ASiteService.per*(..))")
        public Object logExecutionTime(ProceedingJoinPoint joinPoint)
            throws Throwable {
            long start = System.currentTimeMillis();

            Object proceed = joinPoint.proceed();

            long executionTime = System.currentTimeMillis() - start;
            System.out.println(joinPoint.getSignature() + " executed in "
+ executionTime + "ms");

            return proceed;
        }
    }
```

```
public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    public WebAppInitializer() {
        System.out.println("WebAppInitializer created");
    }

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] {
            AppConfig.class,
            AOPConfig.class,
            AOPConfig2.class,
            AOPConfig3.class,
        };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { WebConfig.class };
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

    @Override
    protected Filter[] getServletFilters() {
        CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();
        characterEncodingFilter.setEncoding("UTF-8");
        characterEncodingFilter.setForceEncoding(true);
        return new Filter[] { characterEncodingFilter };
    }
}
```

```
@Controller
public class AOPController {

    ASiteInterface aSiteService;
    TransactionInterface aopService;

    @Autowired
    public AOPController(ASiteInterface aSiteService, TransactionService aopService) {
        this.aSiteService = aSiteService;
        this.aopService = aopService;
    }

    @RequestMapping ("a3.do")
    public void aop3() {
        aSiteService.anotherTask();
        aSiteService.performTask();
        aSiteService.anotherTask();
        aSiteService.performTask();
    }
}
```



```
public interface ASiteInterface {
    public void tour();
    public void order();
    public void performTask();
    public void anotherTask();
}
```

```
@Override
public void performTask() {
    // 실제 비즈니스 로직이 여기에 위치합니다.
    int sum = 0;
    for (int i = 0; i < 1000; i++) {
        sum += i;
    }
    System.out.println(" performTask>> " + sum + " >> =====");
    try {
        Thread.sleep(2000); // 2초간 대기
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

```
@Override
public void anotherTask() {
    // 실제 비즈니스 로직이 여기에 위치합니다.
    int sum = 0;
    for (int i = 0; i < 1000; i++) {
        sum += i;
    }
    System.out.println(" anotherTask>> " + sum + " >> =====");
    try {
        Thread.sleep(1000); // 1초간 대기
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

```
<%@ page contentType="text/html; charset=UTF-8"
    language="java" %>
<html>
<head>
    <title>Title</title>
    <link rel="stylesheet"
        href=" ../resources/css/out2.css">
</head>
<body>
a3 site 화면...
</body>
</html>
```

```
ASiteService.anotherTask() called 1 times, average execution time: 1008ms
performTask>> 499500 >> =====
void com.multi.spring2.aop.service.ASiteService.performTask() executed in 2012ms
anotherTask>> 499500 >> =====
ASiteService.anotherTask() called 2 times, average execution time: 1005ms
performTask>> 499500 >> =====
void com.multi.spring2.aop.service.ASiteService.performTask() executed in 2015ms
```

Transaction

```
선택 관리자: 명령 프롬프트 - mysql -P 3366 -u mega -p
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Administrator.DESKTOP-OK5D7RI>mysql -P 3366 -u mega -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 177
Server version: 8.0.21 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\w' to clear the current input statement.

mysql>
```

```
선택 관리자: 명령 프롬프트 - mysql -P 3366 -u root -p
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Administrator.DESKTOP-OK5D7RI>mysql -P 3366 -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 176
Server version: 8.0.21 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\w' to clear the current input statement.

mysql>
```

```
mysql> create user 'mega'@'localhost' identified by '1234';
mysql> grant all privileges on *.* to 'mega'@'localhost';
```

```
mysql> select @@autocommit;
```

@@autocommit
1

```
1 row in set (0.00 sec)
```

```
mysql> set @@autocommit = 0;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select @@autocommit;
```

@@autocommit
0

```
1 row in set (0.00 sec)
```

autocommit 바꾸기



```
mysql> use shop;
Database changed
mysql> show tables;
+-----+
| Tables_in_shop |
+-----+
| bbs              |
| member          |
| member2         |
| member3         |
| member333       |
| member3333      |
| member4         |
| newtable        |
| reply          |
+-----+
10 rows in set (0.01 sec)

mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
| 1             |
+-----+
1 row in set (0.00 sec)

mysql> insert into member values ('1',' ',' ',' ');
Query OK, 1 row affected (0.05 sec)
```

```
mysql> select * from member;
+----+-----+-----+-----+
| id  | pw    | name  | tel   |
+----+-----+-----+-----+
| 1   |      |      |      |
| a   |      |      |      |
| admin | admin | admin | 111   |
| apple | apple | apple | 222   |
| b   |      |      |      |
| c   |      |      |      |
| computer | test55 | test  | 333   |
| cookie | cookie | cookie | 011   |
| d   |      |      |      |
| music | music | music | 013   |
| phone | phone | phone | 014   |
| spring1000 | spring | spring | spring |
| spring2 | spring | spring | spring |
| win   | win   | win   | 555   |
| winter | spring | spring | spring |
+----+-----+-----+-----+
15 rows in set (0.01 sec)

mysql>
```

```
선택 관리자: 명령 프롬프트 - mysql -P 3366 -u mega -p
mysql> select * from member;
```

id	pw	name	tel
1			
a	admin	admin	111
apple	apple	apple	222
b			
c	computer	test55	333
cookie	cookie	cookie	011
d	music	music	013
phone	phone	phone	014
spring1000	spring	spring	spring
spring2	spring	spring	spring
win	win	win	555
winter	spring	spring	spring

```
15 rows in set (0.00 sec)
mysql>
```

```
mysql> insert into member values ('2','','','');
Query OK, 1 row affected (0.00 sec)

mysql> insert into member values ('3','','','');
Query OK, 1 row affected (0.00 sec)

mysql> select * from member;
```

id	pw	name	tel
1			
2			
3			
admin	admin	admin	111
apple	apple	apple	222
b			
c			
computer	test55	test	333
cookie	cookie	cookie	011
d			
music	music	music	013
phone	phone	phone	014
spring1000	spring	spring	spring
spring2	spring	spring	spring
win	win	win	555
winter	spring	spring	spring

17 rows in set (0.00 sec)

```
mysql> select * from member;
```

id	pw	name	tel
1			
2			
3			
admin	admin	admin	111
apple	apple	apple	222
b			
c			
computer	test55	test	333
cookie	cookie	cookie	011
d			
music	music	music	013
phone	phone	phone	014
spring1000	spring	spring	spring
spring2	spring	spring	spring
win	win	win	555
winter	spring	spring	spring

17 rows in set (0.00 sec)

```
mysql>
```

```
선택 관리자: 명령 프롬프트 - mysql -P 3366 -u root -p
mysql> insert into member values ('4', '', '', '');
Query OK, 1 row affected (0.00 sec)

mysql> insert into member values ('5', '', '', '');
Query OK, 1 row affected (0.00 sec)

mysql> rollback;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from member;
```

id	pw	name	tel
1			
2			
3			
a			
admin	admin	admin	111
apple	apple	apple	222
b			
c			
computer	test55	test	333
cookie	cookie	cookie	011
d			
music	music	music	013
phone	phone	phone	014
spring1000	spring	spring	spring
spring2	spring	spring	spring
win	win	win	555
winter	spring	spring	spring

17 rows in set (0.00 sec)

```
선택 관리자: 명령 프롬프트 - mysql -P 3366 -u mega -p
```

```
mysql> select * from member;
```

id	pw	name	tel
1			
2			
3			
a			
admin	admin	admin	111
apple	apple	apple	222
b			
c			
computer	test55	test	333
cookie	cookie	cookie	011
d			
music	music	music	013
phone	phone	phone	014
spring1000	spring	spring	spring
spring2	spring	spring	spring
win	win	win	555
winter	spring	spring	spring

17 rows in set (0.00 sec)

관리자: 명령 프롬프트 - mysql -P 3366 -u root -p

```
mysql> @@autocommit
+-----+
| @@autocommit |
+-----+
| 1 |
+-----+
1 row in set (0.00 sec)

mysql> set @@autocommit = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> select @@autocommit;
+-----+
| @@autocommit |
+-----+
| 0 |
+-----+
1 row in set (0.00 sec)

mysql> insert into member values('com2',' ',' ',' ');
Query OK, 1 row affected (0.01 sec)

mysql> select * from member;
+----+----+-----+-----+
| id | pw | name | tel |
+----+----+-----+-----+
| com1 |  |  |  |
| com2 |  |  |  |
| kim | kim | kim | 123 |
| lim | lim | lim | 789 |
| park | park | park | 456 |
+----+----+-----+-----+
5 rows in set (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

commit 을 하면 서버에 반영되어 mega에서 확인이 가능

관리자: 명령 프롬프트 - mysql -P 3366 -u mega -p

```
hobby
member
myhobby
reply
tour
6 rows in set (0.00 sec)

mysql> select * from member;
+----+----+-----+-----+
| id | pw | name | tel |
+----+----+-----+-----+
| com1 |  |  |  |
| kim | kim | kim | 123 |
| lim | lim | lim | 789 |
| park | park | park | 456 |
+----+----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from member;
+----+----+-----+-----+
| id | pw | name | tel |
+----+----+-----+-----+
| com1 |  |  |  |
| kim | kim | kim | 123 |
| lim | lim | lim | 789 |
| park | park | park | 456 |
+----+----+-----+-----+
4 rows in set (0.01 sec)

mysql> select * from member;
+----+----+-----+-----+
| id | pw | name | tel |
+----+----+-----+-----+
| com1 |  |  |  |
| com2 |  |  |  |
| kim | kim | kim | 123 |
| lim | lim | lim | 789 |
| park | park | park | 456 |
+----+----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

관리자: 명령 프롬프트 - mysql -P 3366 -u root -p

9 rows in set (0.00 sec)

```
mysql> insert into member values('com7','', '', '', '');  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into member values('com8','', '', '', '');  
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from member;
```

id	pw	name	tel
com1			
com2			
com3			
com4			
com5			
com6			
com7			
com8			
kim	kim	kim	123
lim	lim	lim	789
park	park	park	456

11 rows in set (0.00 sec)

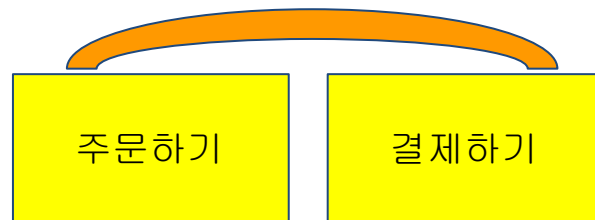
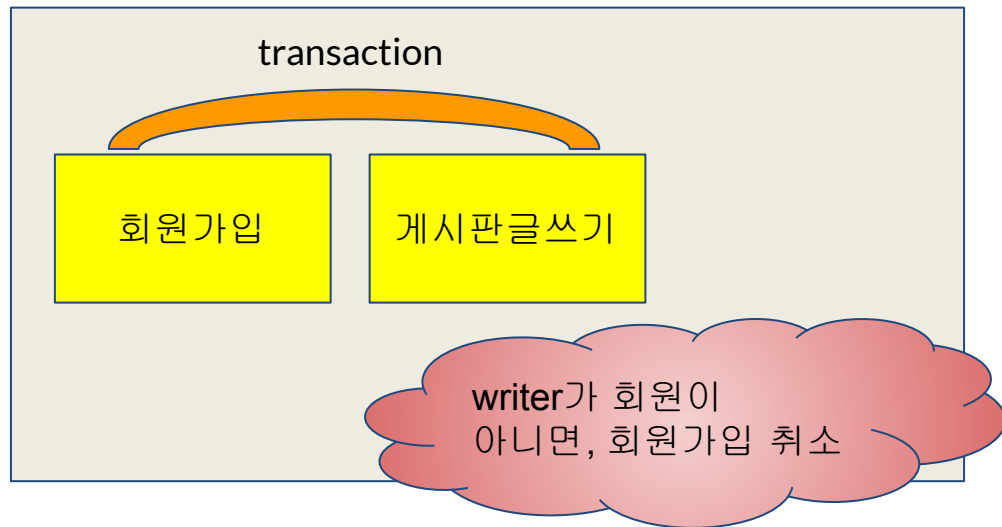
```
mysql> rollback;  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select * from member;
```

id	pw	name	tel
com1			
com2			
com3			
com4			
com5			
com6			
kim	kim	kim	123
lim	lim	lim	789
park	park	park	456

9 rows in set (0.00 sec)

rollback 을 하면 처리 되지 않고 사라지게 된다.
commit 하지 않고 **exit** 하는 경우와 같음.



회원가입 먼저

id:

win55

pw:

....

name:

name

tel:

011

게시판 글쓰기 나중

제목:

title

내용:

content

작성자:

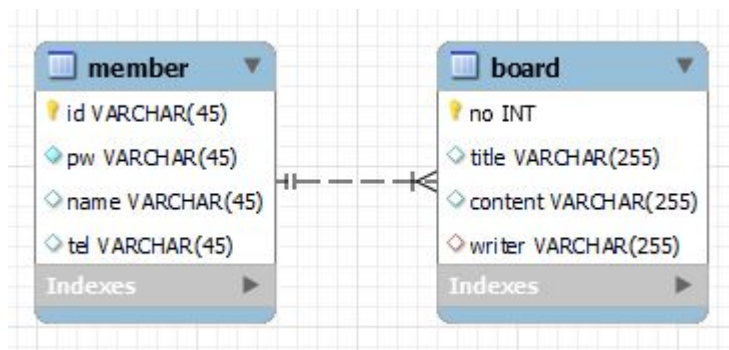
summer10000000000000

글쓰기

```

<div class="container">
  <div class="section">
    <div class="section-header">회원가입 먼저</div>
    <div class="section-content">
      <form action="/transaction.do" method="POST">
        <div class="form-group">
          <label for="id">id:</label>
          <input type="text" id="id" name="id" required value="win55">
        </div>
        <div class="form-group">
          <label for="pw">pw:</label>
          <input type="password" id="pw" name="pw" required value="1234">
        </div>
        <div class="form-group">
          <label for="name">name:</label>
          <input type="text" id="name" name="name" required value="name">
        </div>
        <div class="form-group">
          <label for="tel">tel:</label>
          <input type="text" id="tel" name="tel" value="011">
        </div>
      </form>
    </div>
  </div>
  <div class="section">
    <div class="section-header">게시판 글쓰기 나중</div>
    <div class="section-content">
      <div class="form-group">
        <label for="title">제목:</label>
        <input type="text" id="title" name="title" required value="title">
      </div>
      <div class="form-group">
        <label for="content">내용:</label>
        <input type="text" id="content" name="content" required value="content">
      </div>
      <div class="form-group">
        <label for="writer">작성자:</label>
        <input type="text" id="writer" name="writer" required value="summer10000000000000">
      </div>
      <div class="form-group">
        <input type="submit" value="글쓰기">
      </div>
    </div>
  </div>
</div>

```



```
use shop2;
```

```
CREATE TABLE `member` (  
  `id` varchar(45) NOT NULL,  
  `pw` varchar(45) NOT NULL,  
  `name` varchar(45) DEFAULT NULL,  
  `tel` varchar(45) DEFAULT NULL,  
  PRIMARY KEY (`id`)  
);
```

```
CREATE TABLE `board` (  
  `no` int NOT NULL AUTO_INCREMENT,  
  `title` varchar(255) DEFAULT NULL,  
  `content` varchar(255) DEFAULT NULL,  
  `writer` varchar(255) DEFAULT NULL,  
  PRIMARY KEY (`no`),  
  KEY `writer` (`writer`),  
  CONSTRAINT `board_ibfk_1` FOREIGN KEY (`writer`)  
    REFERENCES `member` (`id`)  
);
```

게시판 글쓰기 나중

제목:

title

내용:

content

작성자:

win55

글쓰기

회원가입 먼저

id:

win55

pw:

name:

name

tel:

011

게시판 글쓰기 나중

제목:

title

내용:

content

작성자:

summer10000000000000

글쓰기

트랜잭션 완료

transaction result >> 모든 db처리 성공!! commit중

트랜잭션 완료

transaction result >> db처리 에러발생!!! rollback중


```
public interface TransactionInterface {

    public int tran(BoardVO boardVO, MemberVO memberVO)
    throws Exception;
}
```

```
@Mapper
public interface AOPMapper {

    int insertMember(MemberVO memberVO);

    int insertBoard(BoardVO boardVO);
}
```

```
<configuration>
    <typeAliases>
        <package
name="com.multi.spring2.aop.domain"/>
    </typeAliases>

</configuration>
```

```
@Service
@Slf4j
public class TransactionService implements TransactionInterface {

    AOPDAO aopDAO;

    @Autowired
    public TransactionService(AOPDAO aopDAO) {
        this.aopDAO = aopDAO;
    }

    @Override
    @Transactional
    public int tran(BoardVO boardVO, MemberVO memberVO) {

        int result = 1;

        // 1. 회원가입 처리
        int memberResult = aopDAO.insertMember(memberVO);

        // 2. 게시물 작성 처리
        int boardResult = aopDAO.insertBoard(boardVO);

        if (memberResult != 1 || boardResult != 1) {
            throw new RuntimeException("error..rolling back.....");
        }
        return result;
    }
}
```



```

@Repository
@Slf4j
public class AOPDAO {

    private final SqlSessionTemplate
    sqlSessionTemplate;

    @Autowired
    public AOPDAO(SqlSessionTemplate
    sqlSessionTemplate) {
        this.sqlSessionTemplate = sqlSessionTemplate;
    }

    public int insertMember(MemberVO memberVO) {
        int rows =
        sqlSessionTemplate.getMapper(AOPMapper.class).insertMember(memberVO);
        return rows;
    }

    public int insertBoard(BoardVO boardVO) {
        int rows =
        sqlSessionTemplate.getMapper(AOPMapper.class).insertBoard(boardVO);
        return rows;
    }
}

```

```

@Controller
public class AOPController {

    ASiteInterface aSiteService;
    TransactionInterface aopService;

    @Autowired
    public AOPController(ASiteInterface aSiteService, TransactionService
    aopService) {
        this.aSiteService = aSiteService;
        this.aopService = aopService;
    }

    @RequestMapping("a.do")
    public void aop1() {
        aSiteService.tour();
    }

    @RequestMapping("transaction.do")
    public String transaction(BoardVO bbsVO2,
        MemberVO memberVO,
        Model model) {

        int result = 0;
        try {
            result = aopService.tran(bbsVO2, memberVO);
            model.addAttribute("result", "모든 db처리 성공!! commit중");
        } catch (Exception e) {
            model.addAttribute("result", "db처리 에러발생!!! rollback중");
            e.printStackTrace();
        }

        System.out.println("result: " + result);

        return "/aop/aop_result";
    }
}

```

```

@Data
@AllArgsConstructor
@NoArgsConstructor
public class BoardVO {
    private int no;           // 게시글의 고유 ID
    private String title;     // 게시글 제목
    private String content;   // 게시글 내용
    private String writer;    // 게시글 작성자
    (member 테이블의 id를 참조)
}

```

```

@Data
@AllArgsConstructor
@NoArgsConstructor
public class MemberVO {
    private String id;
    private String pw;
    private String name;
    private String tel;
}

```

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper
    namespace="com.multi.spring2.aop.mapper.AOPMapper">

    <insert id="insertMember" parameterType="MemberVO">
        insert into member
        values
            (#{id}, #{pw}, #{name}, #{tel})
    </insert>

    <insert id="insertBoard" parameterType="BoardVO">
        insert into board
        values
            (null, #{title}, #{content}, #{writer})
    </insert>

</mapper>

```

1. AOP

2. Advice

- a. Before
- b. After
- c. After Running
- d. Around
- e. After Throwing

3. 핵심 기능, 부가 기능

4. Transaction