Day7; 20220913

날짜	
유형	
태그	

spring/src at main · u8yes/spring

Contribute to u8yes/spring development by creating an account on GitHub.

https://github.com/u8yes/spring/tree/main/src

u8yes/spring

1 Contributor 0 Issues 1 Star 0 Forks

도서의 부록/예제소스를 다운로드

한빛출판네트워크

더 넓은 세상, 더 나은 미래를 위한 아시아 출판 네트워크 :: 한빛미디어, 한빛아카데미, 한빛비즈, 한빛라이프, 한빛에듀

https://www.hanbit.co.kr/support/supplement_survey.html?pcode=B8289488788

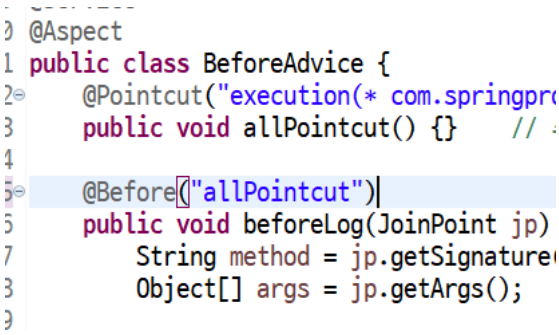
---

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f297200a-8877-4252-aafe-cbb335c604a0/06%EC%9E%A5._Spring%EA%B3%BC_DB.pdf

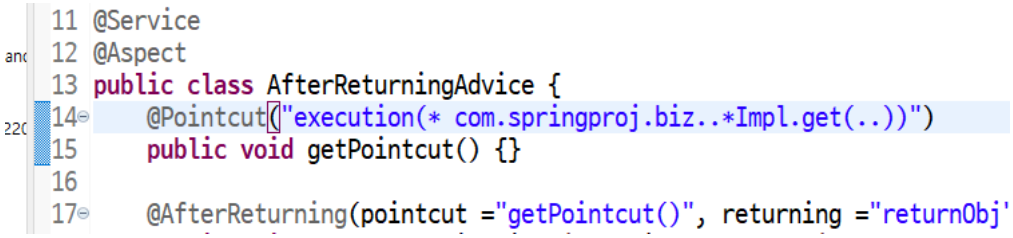
---

오늘의 실수(오타)

@Before("allPointcut()") 에서 소괄호 추상메서드()를 뺐음.



@Pointcut("execution(* com.springproj.biz..*Impl.get*(..))")에서 get뒤에 *을 빼고 썼음



Advice를 annotation으로 작업할 수 있다.

BeforeAdvice

```java
package com.springproj.biz.common;

import org.aspectj.lang.JoinPoint;

@Service
@Aspect
public class BeforeAdvice {
    @Pointcut("execution(* com.springproj.biz..*Impl.*(..))")
    public void allPointcut() {}    // 추상메서드는 반환타입 있다. 일반적으로는 인터페이스 안에 정의할 수 있는 추상메서드.

    @Before("allPointcut()")
    public void beforeLog(JoinPoint jp) {   // 어드바이스
        String method = jp.getSignature().getName();
        Object[] args = jp.getArgs();

        System.out.println("[공통로그] " + method + "(Before) 비즈니스 로직 수행 전 동작");
        System.out.println("args 정보 : " + args[0].toString());
    }
}

```

AfterReturningAdvice

```java
@Aspect
public class AfterReturningAdvice {
    /*
    @Pointcut("execution(* com.springproj.biz..*Impl.get*(..))")
    public void getPointcut() {}

    @AfterReturning(pointcut ="getPointcut()", returning ="returnObj")
    */
    @AfterReturning(pointcut ="PointcutCommon.getPointcut()", returning ="returnObj")

    public void afterLog(JoinPoint jp, Object returnObj) {
        String method = jp.getSignature().getName();

        if(returnObj instanceof BoardVO) {
            BoardVO board = (BoardVO) returnObj;

            if(board.getWriter().equals("민용기")) {
                System.out.println("잘생긴 민용기님이군요.");
            }
        }

        System.out.println("[정상 종료] " + method + "(AfterReturning) 비즈니스 로직 수행 후 동작");
```

AfterThrowAdvice

```java
@Service
@Aspect
public class AfterThrowAdvice {
    @Pointcut("execution(* com.springproj.biz..*Impl.*(..))")
    public void allPointcut() {}

    @AfterThrowing(pointcut = "allPointcut()", throwing = "exception")
    public void exceptionLog(JoinPoint jp, Exception exception) {
        String method = jp.getSignature().getName();

        if(exception instanceof IllegalArgumentException) {
            System.out.println("부적합한 값이 입력되었습니다.");
        }else if(exception instanceof NullPointerException) {
            System.out.println("객체가 존재하지 않습니다.");
        }else {
            System.out.println("문제가 발생하였습니다.");
        }

        System.out.println("[오류발생] " + method + "() 비즈니스 로직 수행 후 동작");
    }
}
```

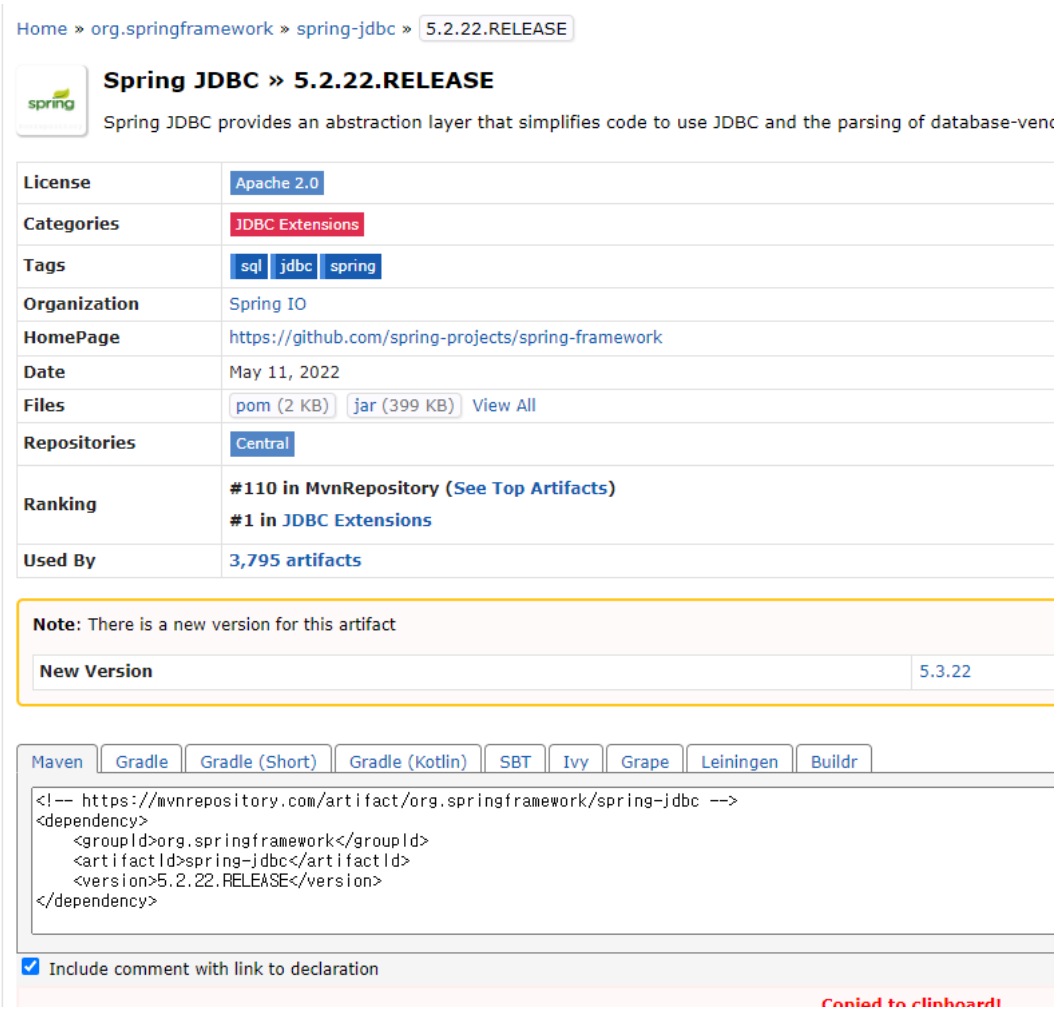
PointcutCommon

```java
package com.springproj.biz.common;

import org.aspectj.lang.annotation.Aspect;

@Aspect
public class PointcutCommon {

    @Pointcut("execution(* com.springproj.biz..*Impl.get*(..))")
    public void getPointcut() {}

    @Pointcut("execution(* com.springproj.biz..*Impl.*(..))")
    public void allPointcut() {}

    @Pointcut("execution(* com.springproj.biz..*Impl.insert*(..))")
    public void insertPointcut() {}

}
```

```
[공통로그] insertService(Before) 비즈니스 로직 수행 전 동작
args 정보 : BoardVO[seq=0, title=첫 게시글, writer=민용기, content=드디어 나는 개발자, date=null, cnt=0]
===> 데이터 저장 성공했습니다^^;
insertService() 메서드 수행에 걸린 시간 : 581(ms)초

[공통로그] getService(Before) 비즈니스 로직 수행 전 동작
args 정보 : 1
getService() 메서드 수행에 걸린 시간 : 37(ms)초

잘생긴 민용기님이군요.
[정상 종료] getService(AfterReturning) 비즈니스 로직 수행 후 동작
BoardVO[seq=1, title=첫 게시글, writer=민용기, content=드디어 나는 개발자, date=2022-09-08, cnt=0]
```

Spring의 JDBC 방법을 편하게

https://mvnrepository.com/artifact/org.springframework/spring-jdbc/5.3.22




```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>5.2.22.RELEASE</version>
</dependency>
```

아래로 변경해줌.

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${org.springframework-version}</version>
    </dependency>
```

```
    <version>${org.springframework-version}</version>
```

동일버전을 다운받게 해줌.

타이핑만 해주면 알아서 변경하게끔

```
    <version>1.0.0-BUILD-SNAPSHOT</version>
    <properties>
        <java-version>11</java-version>
        <org.springframework-version>5.2.22.RELEASE</org.springframework-version>
        <org.aspectj-version>1.6.10</org.aspectj-version>
        <org.slf4j-version>1.6.6</org.slf4j-version>
    </properties>
    <dependencies>
```

https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp/1.4

applicationContext01.xml

```xml
    <context:component-scan base-package="com.springproj.biz"/>

    <!-- AOP annotation을 위한 설정 -->
    <aop:aspectj-autoproxy/>    <!-- Aspect 어노테이션을 적용 -->

    <!-- 스프링 JDBC 설정 -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <!-- setter메서드를 불러주는 프로퍼티, setDriverClassName을 부르기 위해서 name을 정해줌 -->
        <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
        <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"/>
        <property name="username"  value="scott"/>
        <property name="password"  value="tiger"/>
    </bean>

    <!-- springJDBC 기능을 탑재하고 있음 -->
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource" />
    </bean>
```

DB 연결을 한 줄로 다 처리 가능하게 해줌

```java
// CRUD 기능의 메서드 구현
// 글 등록(insert문)
public void insertBoard(BoardVO vo) {   // DTO(DO)
    jdbcTemplate.update(BOARD_INSERT, vo.getTitle(), vo.getWriter(), vo.getContent());  // close까지 함께 같이 처리해줌.
```

```java
// 글 수정(update문)
public void updateBoard(BoardVO vo) {   // DTO(DO)
    jdbcTemplate.update(BOARD_UPDATE, vo.getTitle(), vo.getContent(), vo.getSeq());
```

BoardRowMapper

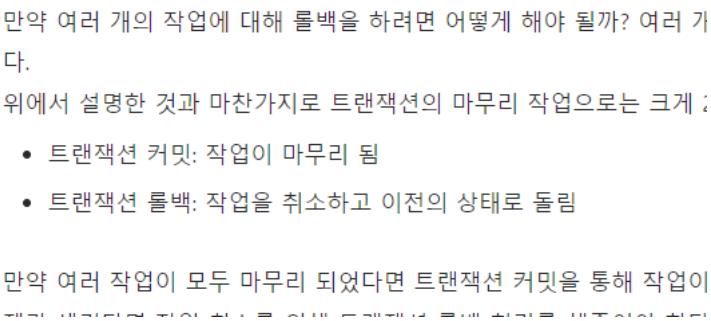
```java
package com.springproj.biz.board.dao;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import com.springproj.biz.board.BoardVO;

public class BoardRowMapper implements RowMapper<BoardVO>{

  @Override
  public BoardVO mapRow(ResultSet rs, int rowNum) throws SQLException { // select문 처리에는 필수 메서드
    BoardVO board = new BoardVO();

    board.setSeq(rs.getInt("seq"));
    board.setTitle(rs.getString("title"));
    board.setWriter(rs.getString("writer"));
    board.setContent(rs.getString("content"));
    board.setRegdate(rs.getDate("regdate"));
    board.setCnt(rs.getInt("cnt"));

    return board;
  }

}
```

BoardDAO

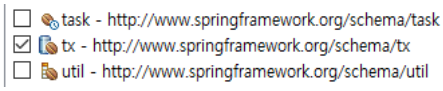
```
package com.springproj.biz.board.dao;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import com.springproj.biz.board.BoardVO;

@Repository("boardDAO")
public class BoardDAO { // DAO(Data Access Object)


  private final String BOARD_INSERT
  = "insert into board(seq, title, writer, content) values (" + "(select nvl(max(seq),0)+1 from board), ?, ?, ?)";  // nvl: Null값이면 0

  private final String BOARD_UPDATE
  = "update board set title = ?, content = ? where seq = ?";

  private final String BOARD_DELETE
  = "delete from board where seq = ?";

  private final String BOARD_SELECT
  = "select * from board where seq = ?";

  private final String BOARD_LIST
  = "select * from board";

  @Autowired
  private JdbcTemplate jdbcTemplate;

  // CRUD 기능의 메서드 구현
  // 글 등록(insert문)
  public void insertBoard(BoardVO vo) { // DTO(DO)
    jdbcTemplate.update(BOARD_INSERT, vo.getTitle(), vo.getWriter(), vo.getContent());  // close까지 함께 같이 처리해줌.


  }

  // 목록보기(select문)
  public BoardVO getBoard(int seq) {
    Object[] args = {seq};

    return jdbcTemplate.queryForObject(BOARD_SELECT, args, new BoardRowMapper());   // 하나의 결과(행)만을 반환 받을 때

  }

  public List<BoardVO> getBoardList() {

    return jdbcTemplate.query(BOARD_LIST, new BoardRowMapper());

  }


  // 글 수정(update문)
  public void updateBoard(BoardVO vo) { // DTO(DO)
    jdbcTemplate.update(BOARD_UPDATE, vo.getTitle(), vo.getContent(), vo.getSeq());


  }

  // 글 삭제(delete문)
  public void deleteBoard(int seq) {

    jdbcTemplate.update(BOARD_UPDATE, seq);
  }
}
```

CRUD = DML

commit은 Database에서 완전히 작업을 마무리시킴.



추가됨




```
xmlns:tx="http://www.springframework.org/schema/tx"

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
```

applicationContext01.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">

  <!-- 스프링이 참조하도록 양식으로 만들어놓은 것, 스피링에 최적화됨 -->
  <context:component-scan base-package="com.springproj.biz"/>

  <!-- AOP annotation을 위한 설정 -->
  <aop:aspectj-autoproxy/>  <!-- Aspect 어노테이션을 적용 -->

  <!-- 스프링 JDBC 설정 방법 -->
  <context:property-placeholder location="classpath:config/database.properties"/> <!-- classpath가 나오면 resource 폴더를 말한다는 것을 인식해

  <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="${driver}" />
    <property name="url" value="${url}"/>
    <property name="username"  value="${name}"/>
    <property name="password"  value="${pw}"/>
  </bean>


  <!-- springJDBC 기능을 탑재하고 있음 -->
```

```
  <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <property name="dataSource" ref="dataSource" />
  </bean>

  <!-- Transaction 설정, Spring이 정의한 것을 이용하는 것 -->
  <bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"></property>
  </bean>

  <tx:advice id="txAdvice" transaction-manager="txManager">
    <tx:attributes>
      <tx:method name="get*" read-only="true"/> <!-- get은 제외시킴 -->
      <tx:method name="*"/>

    </tx:attributes>
  </tx:advice>

  <aop:config>
    <aop:pointcut id="txPointcut" expression="execution(* com.springproj.biz..Impl.*(..))"/>
    <aop:advisor advice-ref="txAdvice" pointcut-ref="txPointcut"/>  <!-- aspect와 똑같은 기능을 구현, 왜냐하면 정확히 지정한 메서드가 없음 -->
  </aop:config>

</beans>
```