



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

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.

```
1 @Aspect
2 public class BeforeAdvice {
3     @Pointcut("execution(* com.springproj.biz..*Impl.get*(..))")
4     public void allPointcut() {} // :
5
6     @Before("allPointcut()")
7     public void beforeLog(JoinPoint jp)
8     {
9         String method = jp.getSignature().getName();
10         Object[] args = jp.getArgs();
11     }
12 }
```

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

```
11 @Service
12 @Aspect
13 public class AfterReturningAdvice {
14     @Pointcut("execution(* com.springproj.biz..*Impl.get*(..))")
15     public void getPointcut() {}
16
17     @AfterReturning(pointcut = "getPointcut()", returning = "returnObj")
18     public void afterReturningAdvice(JoinPoint jp, Object returnObj) {}
19 }
```

Advice를 annotation으로 작업할 수 있다.

BeforeAdvice

```

1 package com.springproj.biz.common;
2
3 import org.aspectj.lang.JoinPoint;
4
5
6
7
8
9 @Service
10 @Aspect
11 public class BeforeAdvice {
12     @Pointcut("execution(* com.springproj.biz..*Impl.*(..))")
13     public void allPointcut() {} // 추상메서드는 반환타입 있다. 일반적으로는 인터페이스 안에 정의할 수 있는 추상메서드.
14
15     @Before("allPointcut()")
16     public void beforeLog(JoinPoint jp) { // 어드바이스
17         String method = jp.getSignature().getName();
18         Object[] args = jp.getArgs();
19
20         System.out.println("[공통로그] " + method + "(Before) 비즈니스 로직 수행 전 동작");
21         System.out.println("args 정보 : " + args[0].toString());
22     }
23 }
24

```

AfterReturningAdvice

```

11 @Aspect
12 public class AfterReturningAdvice {
13     /*
14     @Pointcut("execution(* com.springproj.biz..*Impl.get*(..))")
15     public void getPointcut() {}
16
17     @AfterReturning(pointcut = "getPointcut()", returning = "returnObj")
18     */
19     @AfterReturning(pointcut = "PointcutCommon.getPointcut()", returning = "returnObj")
20
21     public void afterLog(JoinPoint jp, Object returnObj) {
22         String method = jp.getSignature().getName();
23
24         if(returnObj instanceof BoardVO) {
25             BoardVO board = (BoardVO) returnObj;
26
27             if(board.getWriter().equals("민용기")) {
28                 System.out.println("잘생긴 민용기님이군요.");
29             }
30         }
31
32         System.out.println("[정상 종료] " + method + "(AfterReturning) 비즈니스 로직 수행 후 동작");
33     }
34 }

```

AfterThrowAdvice

```

8
9 @Service
10 @Aspect
11 public class AfterThrowAdvice {
12     @Pointcut("execution(* com.springproj.biz.*Impl.*(..))")
13     public void allPointcut() {}
14
15     @AfterThrowing(pointcut = "allPointcut()", throwing = "exception")
16     public void exceptionLog(JoinPoint jp, Exception exception) {
17         String method = jp.getSignature().getName();
18
19         if(exception instanceof IllegalArgumentException) {
20             System.out.println("부적합한 값이 입력되었습니다.");
21         } else if(exception instanceof NullPointerException) {
22             System.out.println("객체가 존재하지 않습니다.");
23         } else {
24             System.out.println("문제가 발생하였습니다.");
25         }
26
27         System.out.println("[오류발생] " + method + "() 비즈니스 로직 수행 후 동작");
28     }
29 }

```

PointcutCommon

```

1 package com.springproj.biz.common;
2
3 import org.aspectj.lang.annotation.Aspect;
4
5
6 @Aspect
7 public class PointcutCommon {
8
9     @Pointcut("execution(* com.springproj.biz.*Impl.get*(..))")
10    public void getPointcut() {}
11
12    @Pointcut("execution(* com.springproj.biz.*Impl.*(..))")
13    public void allPointcut() {}
14
15    @Pointcut("execution(* com.springproj.biz.*Impl.insert*(..))")
16    public void insertPointcut() {}
17
18 }
19

```

```

<terminated> boardrequestclient (5) [Java Application] C:\Program Files\Java\jdk-11.0.10\bin\javaw.exe (2022. 9. 13. 오전 11:58:17 ~ 오전 11:58:19) [pid: 6940]
[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>

Home » org.springframework » spring-jdbc » 5.2.22.RELEASE



Spring JDBC » 5.2.22.RELEASE

Spring JDBC provides an abstraction layer that simplifies code to use JDBC and the parsing of database-ven

License	Apache 2.0
Categories	JDBC Extensions
Tags	sql jdbc spring
Organization	Spring IO
HomePage	https://github.com/spring-projects/spring-framework
Date	May 11, 2022
Files	pom (2 KB) jar (399 KB) View All
Repositories	Central
Ranking	#110 in MvnRepository (See Top Artifacts) #1 in JDBC Extensions
Used By	3,795 artifacts

Note: There is a new version for this artifact

New Version

5.3.22

Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr

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

☒ Include comment with link to declaration

Copied to clipboard

```
<!-- 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>
```

```
</artifact></spring-jdbc>/artifact>
<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>

Indexed Artifacts (29.6M)

Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers

Home » commons-dbcp » commons-dbcp » 1.4

Commons DBCP » 1.4
Commons Database Connection Pooling

License	Apache 2.0
Categories	JDBC Pools
Tags	pooling jdbc pool
HomePage	http://commons.apache.org/dbcp/
Date	Feb 07, 2010
Files	pom (10 KB) jar (156 KB) View All
Repositories	Central Alfresco Apache Releases Apache Staging JBiblio Redhat GA Unvus
Ranking	#220 in MvnRepository (See Top Artifacts) #2 in JDBC Pools
Used By	1,945 artifacts

Maven
Gradle
Gradle (Short)
Gradle (Kotlin)
SBT
Ivy
Grape
Leiningen
Buildr

```
<!-- https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp -->
<dependency>
  <groupId>commons-dbcp</groupId>
  <artifactId>commons-dbcp</artifactId>
  <version>1.4</version>
</dependency>
```

☒ Include comment with link to declaration

applicationContext01.xml

```

12 <context:component-scan base-package="com.springproj.biz"/>
13
14 <!-- AOP annotation을 위한 설정 -->
15 <aop:aspectj-autoproxy/> <!-- Aspect 어노테이션을 적용 -->
16
17 <!-- 스프링 JDBC 설정 -->
18 <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
19     <!-- setter메서드를 불러주는 프로퍼티, setDriverClassName을 부르기 위해서 name을 정해줌 -->
20     <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
21     <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
22     <property name="username" value="scott" />
23     <property name="password" value="tiger" />
24 </bean>
25
26 <!-- springJDBC 기능을 탑재하고 있음 -->
27 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
28     <property name="dataSource" ref="dataSource" />
29 </bean>
30
31

```

DB 연결을 한 줄로 다 처리 가능하게 해줌

```

// 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.
}

```

```

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

```

BoardRowMapper

```

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_DELETE, seq);
    }
}
```


CRUD = DML

commit은 Database에서 완전히 작업을 마무리시킴.

만약 여러 개의 작업에 대해 롤백을 하려면 어떻게 해야 될까? 여러 개다.

위에서 설명한 것과 마찬가지로 트랜잭션의 마무리 작업으로는 크게 :

- 트랜잭션 커밋: 작업이 마무리 됨
- 트랜잭션 롤백: 작업을 취소하고 이전의 상태로 돌림

만약 여러 작업이 모두 마무리 되었다면 트랜잭션 커밋을 통해 작업이
해당 데이터베이스에 반영되는 것을 의미하며, 트랜잭션 롤백을 취소하면 이전의 상태로 돌림

추가됨

- ☐ task - <http://www.springframework.org/schema/task>
- ☒ tx - <http://www.springframework.org/schema/tx>
- ☐ util - <http://www.springframework.org/schema/util>

```
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, 스프링에 최적화된 -->
  <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 말한다라는 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>

```

