



# Spring Framework

✓ 원리를 알면 IT가 맛있다

**Spring Framework for Beginners**

chapter **05.**

---

# Transaction 및 MyBatis 처리.

- 트랜잭션 종류
- 스프링의 트랜잭션 지원 (TransactionManager)
- 트랜잭션 전파(Propagation)
- 트랜잭션 독립성
- 스프링 트랜잭션 처리 방법 ( 선언적 , 명시적 )
- <tx:advice> 이용한 처리 방법
- @Transactional 어노테이션 이용한 처리 방법
- MyBatis 3.X 연동

### ○ Jdbc tx의 처리 – setAutoCommit(false) ,commit, rollback의 실행

```

//***** TX 시작 *****
con.setAutoCommit(false);

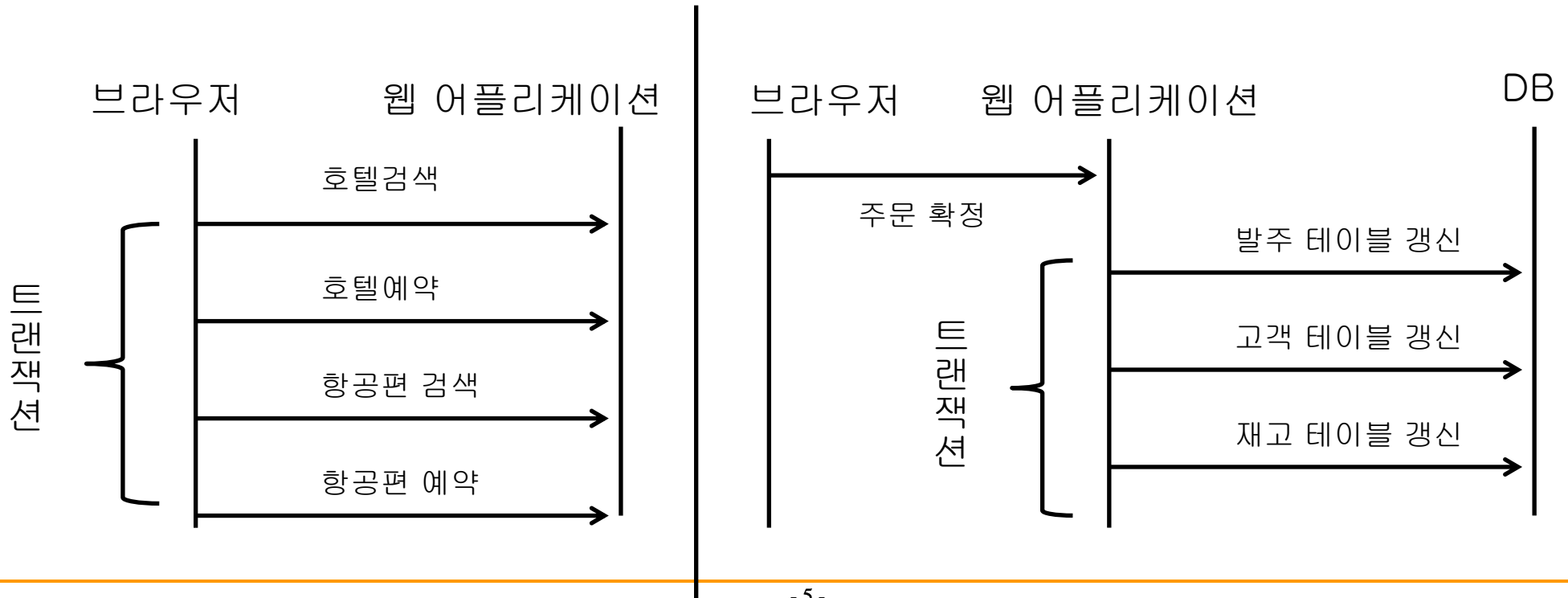
int n = pstmt.executeUpdate();
//int x = 5/0;
if(n==1) con.commit();

}catch(Exception e){

    try {
        con.rollback();
    } catch (SQLException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
}finally{
    try {
```

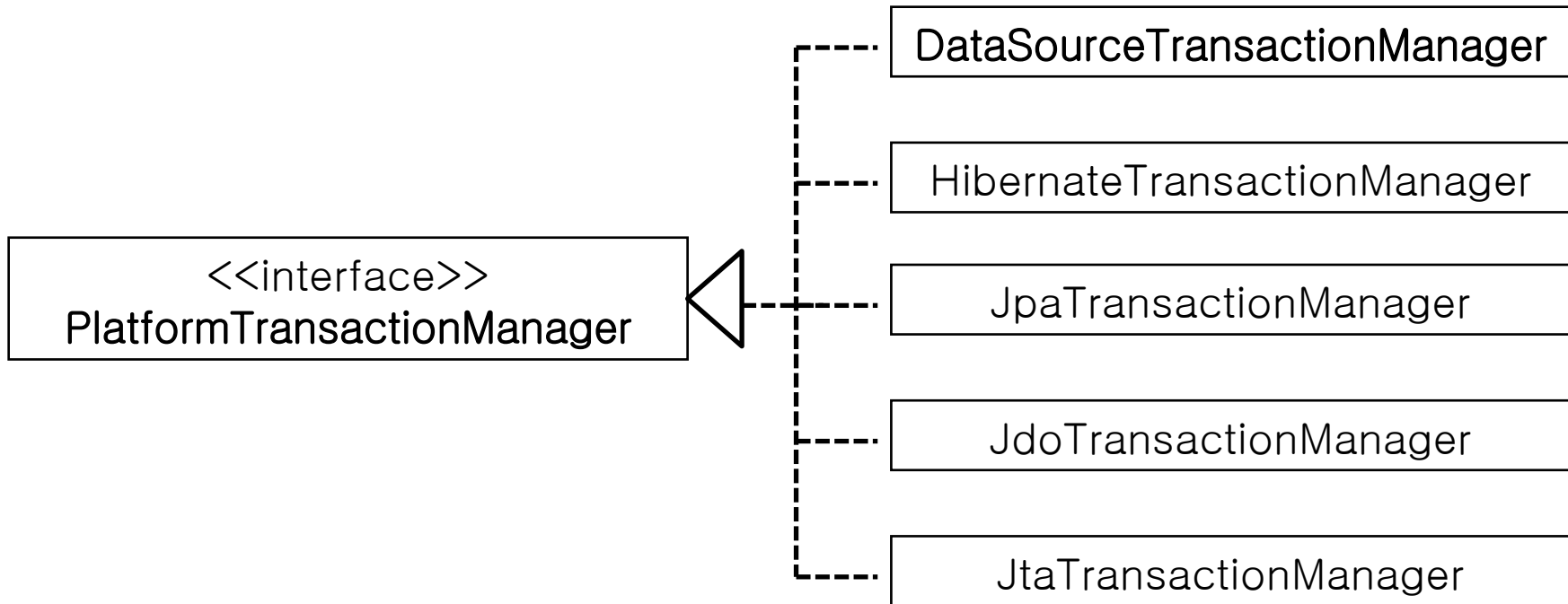
–문제점 개발자가 tx 처리의  
함수 기입 누락등의 오류를  
범할 수 있다.

- 롱 트랜잭션 ( long transaction, loosely coupling transaction )  
: 여러 요청에 걸친 트랜잭션을 의미한다.
- 쇼트 트랜잭션 ( short transaction, tightly coupling transaction )  
: 하나의 요청 안에서 적용되는 트랜잭션을 의미한다.



- 로컬 트랜잭션 ( local transaction )  
: 데이터소스 (일반적으로 데이터베이스)가 하나인 경우.
- 글로벌 트랜잭션 ( global transaction )  
: 데이터소스가 여러 개인 경우.

- 스프링에서는 트랜잭션 관리자를 제공하여 트랜잭션을 처리한다.  
( TransactionManager 클래스 )  
: 트랜잭션의 시작과 종료, 롤백 처리 및 트랜잭션의 정의 정보를 설정.



스프링은 데이터베이스 연동기술과 상관없이 동일한 방법으로 트랜잭션을 처리한다.

### ○ 트랜잭션의 정의 정보 6가지

- 가. 전파(Propagation) 속성
- 나. 독립성 수준
- 다. 시간 만료 ( time out )
- 라. 읽기 전용 상태 ( read-only status )
- 마. 롤백 대상 제외
- 바. 커밋 대상 제외




### ○ 트랜잭션 전파의 이해

예> single transaction ? Or its own transaction ?

```
public class ClientServiceImpl {  
  
    @Autowired  
    private AccountService accountService;  
  
    @Transactional  
    public void updateClient(){  
  
        ..  
        this.accountService.update();  
    }  
}
```

```
public class AccountServiceImpl  
{  
    @Transactional  
    public void update(){  
        ...  
    }  
}
```



### ○ REQUIRED

: 기본값

: 메소드를 수행하는데 tx가 필요하다는 것을 의미한다. 현재 진행중인 tx가 존재하면 해당 tx를 사용하고, 존재하지 않으면 새로운 tx를 생성한다.

### ○ MANDATORY

: 메소드를 수행하는데 tx가 필요하다는 것을 의미한다. REQUIRED와 다르게 진행 중인 tx가 없으면 예외 발생된다.

### ○ REQUIRES\_NEW

: 항상 새로운 tx를 시작한다. 기존 tx가 존재하면 일시중지하고 새로운 tx를 시작한다. 새로운 tx가 종료된 후에 기존 tx가 처리된다.

### ○ SUPPORTS

: 메소드가 tx를 필요로 하지는 않지만 존재할 경우에 tx를 사용한다는 의미이다. 진행 중인 tx가 없어도 메소드는 정상 동작한다.

( tx가 있으면 사용하고 없어도 무관 )

- NOT\_SUPPORTS

: 메소드가 tx이 필요없음을 의미한다. 진행중인 tx가 있으면 잠시 중지하고 메소드가 종료된 후에 tx가 진행된다.

- NEVER

: 메소드가 tx를 절대 사용하지 않음을 의미한다. 진행중인 tx가 있으면 에러 발생.

- NESTED

: 기존 tx가 존재하면 중첩되게 tx 처리됨. 기존 tx가 존재하지 않으면 REQUIRED와 동일하게 동작됨. ( JDBC 3.X 지원 )

- 트랜잭션 처리가 병행해서 실행될 때 각 트랜잭션의 독립성을 결정하는 것을 의미한다.

예> tx1이 데이터베이스의 레코드를 갱신하고 아직 commit 전상태.

이때 tx2가 tx1이 갱신한 레코드를 읽어오려고 한다.

이 경우에 tx2가 갱신된 데이터를 읽어와도 되는가? 하는 문제가 발생할수 있으며 이를 모순되지 않게 처리하는 속성이 독립성이다.

- ISOLATION\_READ\_COMMITTED  
: 다른 tx이 변경했지만 아직 커밋하지 않은 데이터는 읽을 수 없다.
- ISOLATION\_READ\_UNCOMMITTED  
: 다른 tx이 변경하고 아직 커밋하지 않은 데이터를 읽을 수 있다.
- ISOLATION\_REPEATABLE\_READ  
: tx내에서 여러 번 데이터를 읽을 때, 다른 트랜잭션이 도중에 데이터를 갱신해도 같은 값을 읽어온다.
- ISOLATION\_SERIALIZABLE  
: tx를 하나씩 순서대로 처리한다.
- ISOLATION\_DEFAULT  
: 데이터베이스가 제공하는 기본 독립성 수준을 이용한다.

- 시간만료  
: tx이 취소되는 시간 만료 시간을 초단위로 설정한다.
- 읽기 전용 상태  
: tx내의 처리가 읽기전용인지 설정한다. 설정에 의해 DB나 ORM프레임워크 쪽에서 최적화가 이루어진다.
- 롤백 대상 예외  
: 어떤 예외가 발생되었을 때 롤백할 지 설정할 수 있다.  
기본으로는 Runtime예외가 발생되면 자동 롤백 되고, checked 예외는 발생되도 롤백되지 않는다.
- 커밋 대상 예외  
: 어떤 예외가 발생되었을 때 커밋할지 설정할 수 있다. 기본으로는 checked 예외가 발생될 때 자동 커밋된다.

- 명시적 트랜잭션 처리 방법

- 선언적 트랜잭션 처리 방법

: 가장 많이 사용되는 방법으로, 권장되는 처리법이다.

가. <tx:advice> 이용

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>
```

```
<tx:advice id="transactionService">
  <tx:attributes>
    <tx:method name="insert*"
      propagation="REQUIRED"
      isolation="DEFAULT"
      read-only="false"
      timeout="-1"/>
  </tx:attributes>
</tx:advice>
```

```
<aop:config>
  <aop:advisor advice-ref="transactionService" pointcut="execution(* *.insert*(..))"/>
</aop:config>
```

## ○ Pom.xml

```
<!--  
https://mvnrepository.com/artifact/org.springframework/spring-  
jdbc -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-jdbc</artifactId>  
    <version>4.3.2.RELEASE</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.apache.commons/commons-  
dbcp2 -->  
<dependency>  
    <groupId>org.apache.commons</groupId>  
    <artifactId>commons-dbcp2</artifactId>  
    <version>2.5.0</version>  
</dependency>  
<!-- https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc6  
-->  
<dependency>  
    <groupId>com.jslsolucoes</groupId>  
    <artifactId>ojdbc6</artifactId>  
    <version>11.2.0.1.0</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.springframework/spring-  
tx -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-tx</artifactId>  
    <version>4.3.2.RELEASE</version>  
</dependency>
```

## //config.xml

Select XSD namespaces to use in the configuration file

- ☒ aop - <http://www.springframework.org/schema/aop>
- ☒ beans - <http://www.springframework.org/schema/beans>
- ☐ c - <http://www.springframework.org/schema/c>
- ☐ cache - <http://www.springframework.org/schema/cache>
- ☒ context - <http://www.springframework.org/schema/context>
- ☐ jdbc - <http://www.springframework.org/schema/jdbc>
- ☐ jee - <http://www.springframework.org/schema/jee>
- ☐ lang - <http://www.springframework.org/schema/lang>
- ☐ p - <http://www.springframework.org/schema/p>
- ☐ task - <http://www.springframework.org/schema/task>
- ☒ tx - <http://www.springframework.org/schema/tx>
- ☐ util - <http://www.springframework.org/schema/util>



## ○ Config.xml : jdbcTemplate, DataSourceTransactionManager

```
<!-- JDBC 연동 -->
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="location">
        <value>classpath:db.properties</value>
    </property>
</bean>
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource"
    destroy-method="close">
    <property name="driverClassName" value="${driver}" />
    <property name="url" value="${url}" />
    <property name="username" value="${username}" />
    <property name="password" value="${password}" />
</bean>
<!-- DataSourceTransactionManager 설정 -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
</bean>

<!-- transaction-manager="transactionManager" 생략 가능 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager" >
<tx:attributes>
    <tx:method name="add*"
        propagation="REQUIRED"
        isolation="DEFAULT"
        read-only="false"
        timeout="-1"/>
</tx:attributes>
</tx:advice>
```

### ○ Config.xml : jdbcTemplate, DataSourceTransactionManager

```
<aop:config>
  <aop:pointcut expression="execution(* com.service.*Service.*(..))" id="xxx"/>
  <aop:advisor advice-ref="txAdvice" pointcut-ref="xxx"/>
</aop:config>
<!-- DataSourceTransactionManager 설정 -->
```

```
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <property name="dataSource" ref="dataSource" />
</bean>
<!-- JDBC 연동 -->
<bean id="deptDAO" class="com.dao.ProductDAO">
  <property name="jdbcTemplate" ref="jdbcTemplate" />
</bean>
<bean id="deptService" class="com.service.ProductService">
  <property name="dao" ref="deptDAO" />
</bean>
</beans>
```

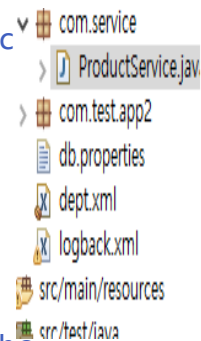
```
public void insert(int deptno, String dname, String loc) {  
  
    String sql = "insert into dept ( deptno, dname, loc ) values ( ? , ? , ? )";  
  
    int n = jdbcTemplate.update(sql, deptno, dname, loc);  
  
    int num = 5/0; // RuntimeException 발생으로 insert 작업은 롤백 된다.  
  
    String sql2 = "update dept set  dname= '가가가' , loc = '제주' where deptno = ?";  
    jdbcTemplate.update(sql2 , deptno);  
  
} // end insert
```

### 나. @Transactional 어노테이션 이용

```
<bean id="transactionManager"  
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
    <property name="dataSource" ref="dataSource" />  
</bean>  
  
<tx:annotation-driven transaction-manager="transactionManager"/>
```

```
@Transactional  
public void insert(int deptno, String dname, String loc){  
    dao.insert(deptno, dname, loc);  
}
```

```
<!--  
https://mvnrepository.com/artifact/org.springframework/s  
pring-jdbc -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-jdbc</artifactId>  
    <version>4.3.2.RELEASE</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.apache.commons/c  
ommons-dbcp2 -->  
<dependency>  
    <groupId>org.apache.commons</groupId>  
    <artifactId>commons-dbcp2</artifactId>  
    <version>2.5.0</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc  
6 -->  
<dependency>  
    <groupId>com.jslsolucoes</groupId>  
    <artifactId>ojdbc6</artifactId>  
    <version>11.2.0.1.0</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.springframework/s  
pring-tx -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-tx</artifactId>  
    <version>4.3.2.RELEASE</version>  
</dependency>
```



```
//특값속업 값속업값번드업
```

```
@Transactional
```

```
public void addOrder(String pcode, int quantity) throws Exception{  
    dao.addOrder(pcode, quantity);  
}
```

# □ @Transactional 어노테이션 이용

Spring Framework

```
com.service
> ProductService.java 41 //특삭열 삭제열 삭제열
42
43 @Transactional
44 public void addOrder(String pcode, int quantity) throws Exception{
45     dao.addOrder(pcode, quantity);
46 }
47
```

- 1) service메소드에 @Transactional 지정 (tx처리 함수)
- 2) Properties - dataSource생성 - transactionManager생성
- 3) Tx-annotation-driven 에 transactionManager 지정

<input checked="" type="checkbox"/>		aop - <a href="http://www.springframework.org/schema/aop">http://www.springframework.org/schema/aop</a>
<input checked="" type="checkbox"/>		beans - <a href="http://www.springframework.org/schema/beans">http://www.springframework.org/schema/beans</a>
<input type="checkbox"/>		c - <a href="http://www.springframework.org/schema/c">http://www.springframework.org/schema/c</a>
<input type="checkbox"/>		cache - <a href="http://www.springframework.org/schema/cache">http://www.springframework.org/schema/cache</a>
<input checked="" type="checkbox"/>		context - <a href="http://www.springframework.org/schema/context">http://www.springframework.org/schema/context</a>
<input type="checkbox"/>		jdbc - <a href="http://www.springframework.org/schema/jdbc">http://www.springframework.org/schema/jdbc</a>
<input type="checkbox"/>		jee - <a href="http://www.springframework.org/schema/jee">http://www.springframework.org/schema/jee</a>
<input type="checkbox"/>		lang - <a href="http://www.springframework.org/schema/lang">http://www.springframework.org/schema/lang</a>
<input type="checkbox"/>		mvc - <a href="http://www.springframework.org/schema/mvc">http://www.springframework.org/schema/mvc</a>
<input type="checkbox"/>		p - <a href="http://www.springframework.org/schema/p">http://www.springframework.org/schema/p</a>
<input type="checkbox"/>		task - <a href="http://www.springframework.org/schema/task">http://www.springframework.org/schema/task</a>
<input checked="" type="checkbox"/>		tx - <a href="http://www.springframework.org/schema/tx">http://www.springframework.org/schema/tx</a>

```
<!-- JDBC 연동 -->
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="location">
        <value>classpath:db.properties</value>
    </property>
</bean>
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource"
    destroy-method="close">
    <property name="driverClassName" value="${driver}" />
    <property name="url" value="${url}" />
    <property name="username" value="${username}" />
    <property name="password" value="${password}" />
</bean>
```

- 1) service메소드에 @Transactional 지정 (tx처리 함수)
- 2) Properties -dataSource생성-transactonManager생성
- 3) Tx-annotation-driven 에 transactonManager 지정

```
<!-- DataSourceTransactionManager 설정 -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" /> </bean>

  <tx:annotation-driven transaction-manager="transactionManager" proxy-target-class="false"/>
<!-- DataSourceTransactionManager 설정 -->

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

<!-- JDBC 연동 -->
<bean id="deptDAO" class="com.dao.ProductDAO">
  <property name="jdbcTemplate" ref="jdbcTemplate" />
</bean>
<bean id="deptService" class="com.service.ProductService">
  <property name="dao" ref="deptDAO" />
</bean>
</beans>
```

- 스프링과의 연동을 위해서 myBatis-spring.jar 라이브러리를 제공한다. ( build path 필요 )
- SqlSessionFactory 빈과 SqlSessionTemplate을 이용한 SqlSession 객체를 얻는다.

```
<!-- MyBatis -->
<bean id="sqlSessionFactory"
      class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="mapperLocations" value="classpath:com/config/testMapper.xml" />
</property>
</bean>
<!-- MyBatis Template-->
<bean id="sqlSession"
      class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg ref="sqlSessionFactory" />
</bean>

<bean id="testDAO" class="com.dao.TestDAO">
  <property name="sessionTemplate" ref="sqlSession" />
</bean>
```



```
SqlSessionTemplate sessionTemplate;

public void setSessionTemplate(SqlSessionTemplate sessionTemplate) {
    this.sessionTemplate = sessionTemplate;
}

// insert
public void insert(TestDTO dto) {

    int n = sessionTemplate.insert("newrecord", dto);
}
```

## ❑ MyBatis 3.x 연동-6개의 레포지토리 pom.xml

*Spring Framework*

```
<!--  
https://mvnrepository.com/artifact/org.mybatis/mybatis -  
-->  
<dependency>  
    <groupId>org.mybatis</groupId>  
    <artifactId>mybatis</artifactId>  
    <version>3.4.6</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.mybatis/mybatis-  
spring -->  
<dependency>  
    <groupId>org.mybatis</groupId>  
    <artifactId>mybatis-spring</artifactId>  
    <version>1.3.2</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.springframework/s  
pring-jdbc -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-jdbc</artifactId>  
    <version>5.0.8.RELEASE</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.apache.commons/co  
mmons-dbcp2 -->  
<dependency>  
    <groupId>org.apache.commons</groupId>  
    <artifactId>commons-dbcp2</artifactId>  
    <version>2.5.0</version>  
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/com.jslsolucoes/ojdbc  
-->  
<dependency>  
    <groupId>com.jslsolucoes</groupId>  
    <artifactId>ojdbc6</artifactId>  
    <version>11.2.0.1.0</version>  
</dependency>  
  
<!--  
https://mvnrepository.com/artifact/org.springframework/spring  
tx -->  
<dependency>  
    <groupId>org.springframework</groupId>  
    <artifactId>spring-tx</artifactId>  
    <version>4.3.20.RELEASE</version>  
</dependency>  
<!--  
https://mvnrepository.com/artifact/org.aspectj/aspectjweaver  
-->  
<dependency>  
    <groupId>org.aspectj</groupId>  
    <artifactId>aspectjweaver</artifactId>  
    <version>1.9.2</version>  
</dependency>
```

- ☒ aop - <http://www.springframework.org/schema/aop>
- ☒ beans - <http://www.springframework.org/schema/beans>
- ☐ c - <http://www.springframework.org/schema/c>
- ☐ cache - <http://www.springframework.org/schema/cache>
- ☒ context - <http://www.springframework.org/schema/context>
- ☐ jdbc - <http://www.springframework.org/schema/jdbc>
- ☐ jee - <http://www.springframework.org/schema/jee>
- ☐ lang - <http://www.springframework.org/schema/lang>
- ☐ mybatis-spring - <http://mybatis.org/schema/mybatis-spring>
- ☐ p - <http://www.springframework.org/schema/p>
- ☐ task - <http://www.springframework.org/schema/task>
- ☒ tx - <http://www.springframework.org/schema/tx>
- ☐ util - <http://www.springframework.org/schema/util>

```
1 <context:annotation-config></context:annotation-config>
2 <!-- 1.jdbc.properties등록 -->
3 <context:property-placeholder location="classpath:com/config/jdbc.properties"/>
4 <!-- 2. dbcp2 생성 :jdbc.properties 이용-->
5 <bean id="myDataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
6     <property name="driverClassName" value="${jdbc.driver}"></property>
7     <property name="url" value="${jdbc.url}"></property>
8     <property name="username" value="${jdbc.userid}"></property>
9     <property name="password" value="${jdbc.passwd}"></property>
10 </bean>
```

```

1 <!-- 3.SqlSessionFactoryBean : SqlSessionFactoryBean기능 -->
2 <bean id="mySqlSessonFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
3     <property name="dataSource" ref="myDataSource"></property>
4     <property name="mapperLocations">
5         <list>
6             <value>classpath:com/config/DeptMapper.xml</value>
7         </list>
8     </property>
9     <property name="typeAliases">
10        <list>
11            <value>com.dto.DeptDTO</value>
12        </list>
13    </property>
14 </bean>
15
16 <!-- 4. SqlSessionTemplate : SqlSession기능 -->
17 <bean id="mySqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
18     <constructor-arg name="sqlSessionFactory" ref="mySqlSessonFactory"></constructor-arg>
19 </bean>
20
21 <!--빈생성 -->
22 <bean id="deptDAO" class="com.dao.DeptDAO"></bean>
23 <bean id="deptService" class="com.service.DeptService"/>
24 </beans>

```

```

1
2 import org.apache.ibatis.type.Alias;
3
4 @Alias("DeptDTO")
5 public class DeptDTO {
6     private int deptno;
7     private String dname;
8     private String loc;
9 }

```

## □ MyBatis 3.x 연동-tx의 처리

- ☒ aop - <http://www.springframework.org/schema/aop>
- ☒ beans - <http://www.springframework.org/schema/beans>
- ☐ c - <http://www.springframework.org/schema/c>
- ☐ cache - <http://www.springframework.org/schema/cache>
- ☒ context - <http://www.springframework.org/schema/context>
- ☐ jdbc - <http://www.springframework.org/schema/jdbc>
- ☐ jee - <http://www.springframework.org/schema/jee>
- ☐ lang - <http://www.springframework.org/schema/lang>
- ☐ mybatis-spring - <http://mybatis.org/schema/mybatis-spring>
- ☐ p - <http://www.springframework.org/schema/p>
- ☐ task - <http://www.springframework.org/schema/task>
- ☒ tx - <http://www.springframework.org/schema/tx>
- ☐ util - <http://www.springframework.org/schema/util>

### Config.xml

```
<!-- JDBC 연동 -->
<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
  <property name="location">
    <value>classpath:com/config/db.properties</value>
  </property>
</bean>

<!-- DataSourceTransactionManager 설정 -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager" proxy-target-class="false"/>
<!-- DataSourceTransactionManager 설정 -->
```

```

8
9<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" |
0    destroy-method="close">
1    <property name="driverClassName" value="${driver}" />
2    <property name="url" value="${url}" />
3    <property name="username" value="${username}" />
4    <property name="password" value="${password}" />
5</bean>
6<!-- mybatis -->
7<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
8    <property name="dataSource" ref="dataSource" />
9    <property name="mapperLocations">
0        <list>
1            <value>classpath:com/config/productMapper.xml</value>
2            <value>classpath:com/config/orderMapper.xml</value>
3        </list>
4    </property>
5    <!-- <property name="mapperLocations" value="classpath:com/dao/*.xml" /> -->
6    <!-- <property name="typeAliasesPackage" value="com.test" /> -->
7</bean>
8
9<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" >
0    <constructor-arg ref="sqlSessionFactory" />
1</bean>
2
3
4<bean id="deptDAO" class="com.dao.ProductDAO">
5    <property name="template" ref="sqlSession" />
6</bean>
7<bean id="deptService" class="com.service.ProductService">
8    <property name="dao" ref="deptDAO" />
9</bean>
10
11

```

```
31     }  
32     //특삭엿엿  곱엿엿 곱엿엿  
33  
34     @Transactional  
35     public void addOrder(String pcode, int quantity) throws Exception{  
36         dao.addOrder(pcode, quantity);  
37     }  
38
```

- 트랜잭션 종류
- 스프링의 트랜잭션 지원 (TransactionManager)
- 트랜잭션 전파(Propagation)
- 트랜잭션 독립성
- 스프링 트랜잭션 처리 방법 ( 선언적 , 명시적 )
- <tx:advice> 이용한 처리 방법
- @Transactional 어노테이션 이용한 처리 방법





**Thank you**

---