[Spring]스프링과트랜잭션

트랜잭션

트랜잭션
물리적으로는 여러 개의 작업이지만 논리적으로는 하나의 작업
이체 = 인출 + 입금
인출 성공 + 입금 실패 --> 이체 실패
다른 예 : 인터넷 상품 구매. 주문 + 결제
트랜잭션의 특징 : ACID
원자성(Atomicity)
일관성(Consistency)
독립성(Isolation)
지속성(Durability)
트랜잭션 격리 수준(Isolation Level)
Read UnCommitted
Read Committed
Repeatable Read
Serializable

Spring 에서 사용가능한 트랜잭션 처리 방법
Hibernate, JDBC, JTA, JDO, JPA 등
이중 JTA는 JTA를 지원하는 컨테이너와 같이 통합 사용되기 때문에 Spring 자체에서 관리 불가

선언적 트랜잭션 처리
XML 설정 파일 이용
AOP 기법 이용
TransactionManager가 하나의 Advisor로 동작함.

선언적 트랜잭션 처리 방법
네임스페이스 설정

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
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-3.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">
```

DataSource 설정
BasicDataSource Bean 등록
driverClassName, url, username, password 등록

TransactionManager 설정
DataSourceTransactionManager Bean 등록
트랜잭션 정책 설정
<tx:advice /> 엘리먼트 이용

```
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
<property name="dataSource" ref="dataSource"/>
</bean>
<tx:advice id="txAdvice" transaction-manager="txManager">
<tx:attributes>
<tx:method name="checkout" propagation="REQUIRED"/>
<tx:method name="get*" read-only="true"/>
</tx:attributes>
</tx:advice>
```

get으로 시작하는 메서드는 모두 읽기 전용
checkout 메서드는 트랜잭션 처리됨. Required!

트랜잭션 정책 설정 - 이어서

<tx:advice /> 엘리먼트의 속성

속성	필수여부	기본값	설명
name	필수		트랜잭션이 적용될 메소드명을 지정한다. 와일드카드(*)를 사용하여 같은 트랜잭션 속성을 하나이상의 메소드와 연관시킬 수 있다. 예를 들어 get*, insert*, find*, delete*, update* 등으로 지정할 수 있다.
propagation	선택	REQUIRED	트랜잭션 전파 방법을 설명한다.
isolation	선택	DEFAULT	트랜잭션 격리 레벨. 데이터베이스가 지원하는 기본 값으로 설정된다.
timeout	선택	-1	트랜잭션 타임아웃 값 (초단위 설정)
read-only	선택	false	트랜잭션이 읽기만 가능한가?
rollback-for	선택		트랜잭션을 Rollback할 예외 타입을 지정한다. 콤마로 구분하여 여러 개를 나열 할 수 있다. 예를 들어 com.MyException, com.AnotherException 등으로 설정한다.
no-rollback-for	선택		트랜잭션을 Rollback하지 않을 예외 타입을 지정한다.

AOP 를 이용한 트랜잭션 적용

<tx:advice /> 는 Advice 등록만 수행함.
실제로 트랜잭션을 적용하기 위해서는 AOP를 사용해야 함.

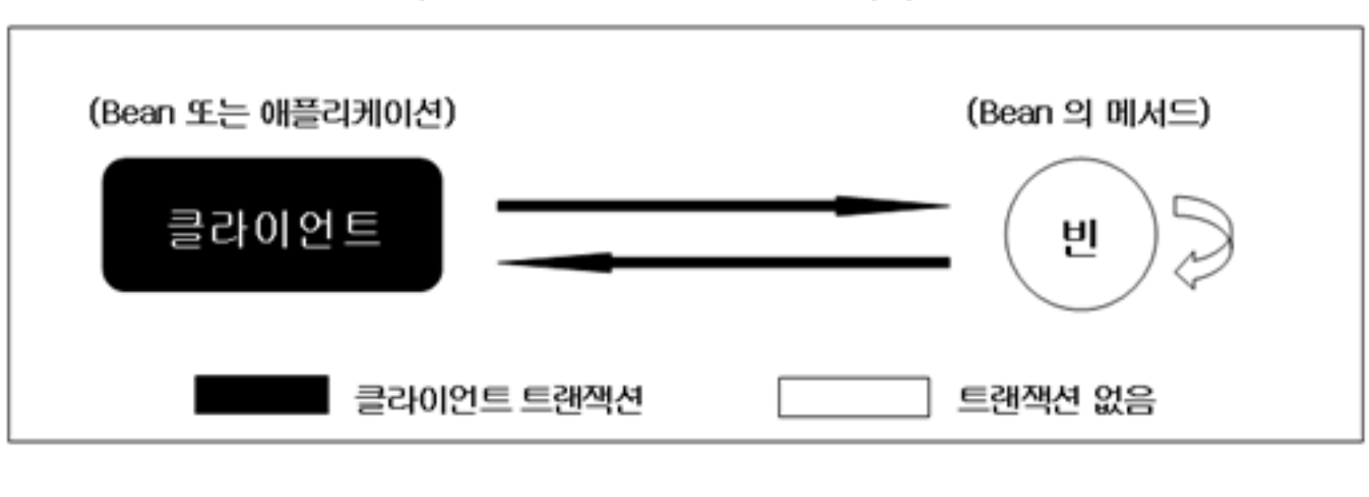
```
<tx:advice id="txAdvice" transaction-manager="txManager">
<tx:attributes>
<tx:method name="checkout" propagation="REQUIRED"/>
<tx:method name="get*" read-only="true"/>
</tx:attributes>
</tx:advice>
<aop:config>
<aop:pointcut id="transactionMethod"
expression="execution(* com.multicampus.biz..*Service.*(..))"/>
<aop:advisor advice-ref="txAdvice" pointcut-ref="transactionMethod"/>
</aop:config>
```

<tx:advice />의 propagation 속성

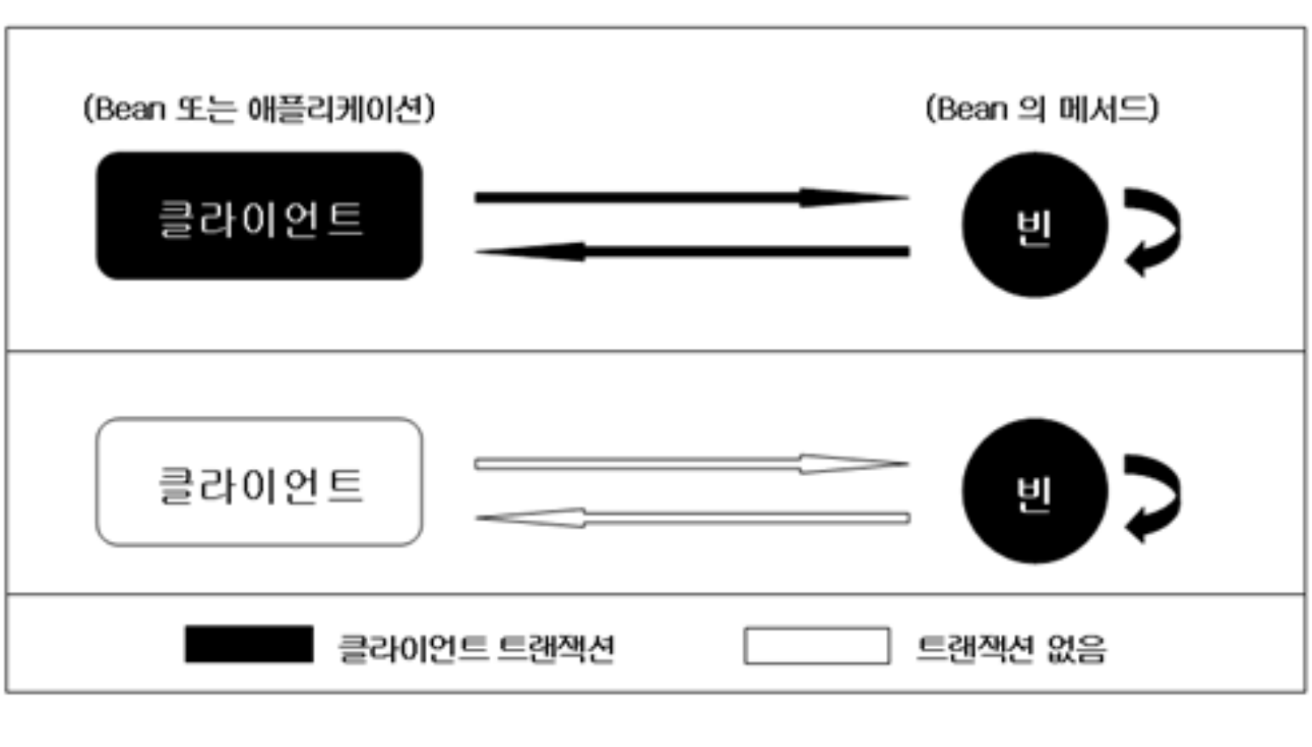
트랜잭션 전파 규칙을 정의함.

전파 방법	설명
REQUIRED	기존의 트랜잭션이 존재하면 같은 트랜잭션에서 실행되며 기존의 트랜잭션이 존재하지 않으면 새로운 트랜잭션을 발생시킨다.
SUPPORTS	기존의 트랜잭션이 존재하면 같은 트랜잭션에서 실행되며 기존의 트랜잭션이 존재하지 않으면 트랜잭션 없이 실행된다.
MANDATORY	기존의 트랜잭션이 존재하면 같은 트랜잭션에서 실행되며 기존의 트랜잭션이 기존의 트랜잭션이 존재하지 않으면 예외가 발생한다.
REQUIRES_NEW	기존의 트랜잭션이 존재하면 실행을 보류시키고 새로운 트랜잭션을 생성시켜 실행된다.
NOT_SUPPORTED	기존의 트랜잭션이 존재해도 트랜잭션 없이 실행된다.
NAVER	기존의 트랜잭션이 존재하면 예외를 발생시키고 기존의 트랜잭션이 존재하지 않으면 트랜잭션 없이 실행된다.
NESTED	기존의 트랜잭션이 존재하면 중첩된 트랜잭션이 실행되며 기존의 트랜잭션이 존재하지 않으면 REQUIRED와 동일하게 실행된다.

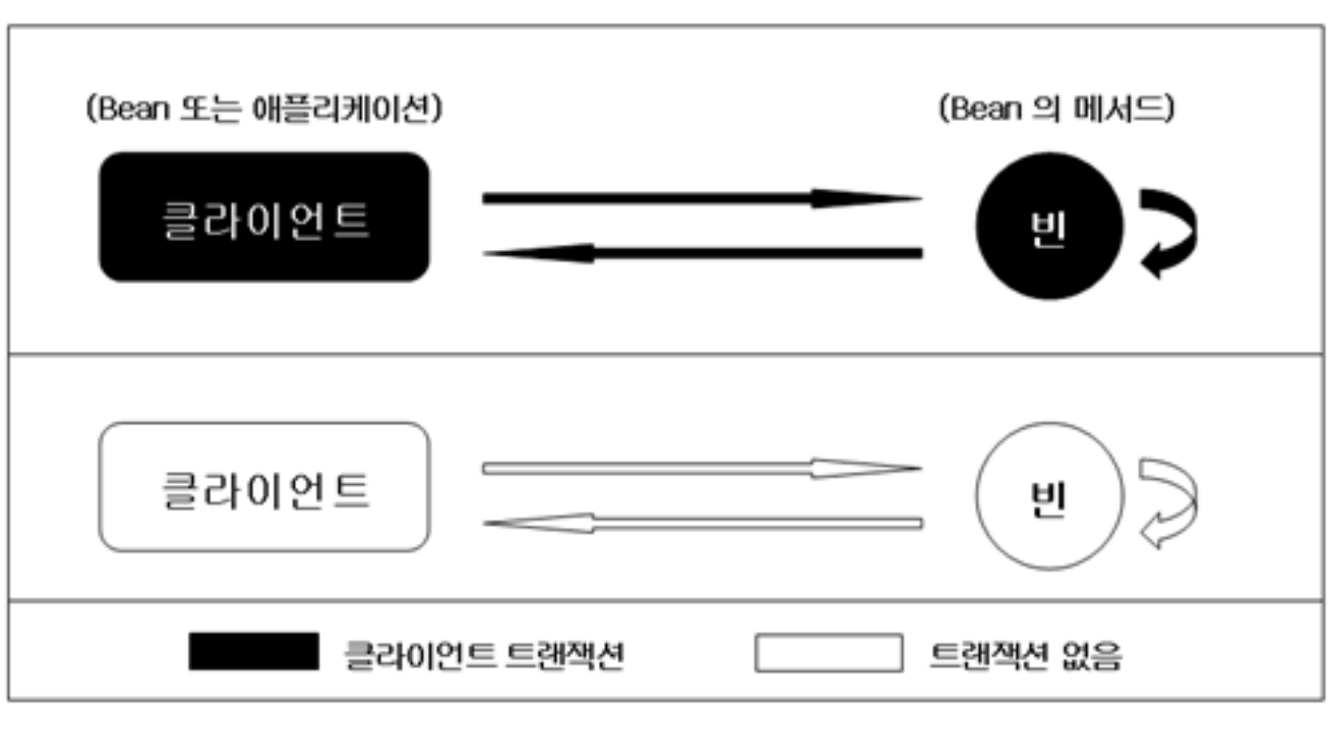
Transaction Propagation (Not Supported)



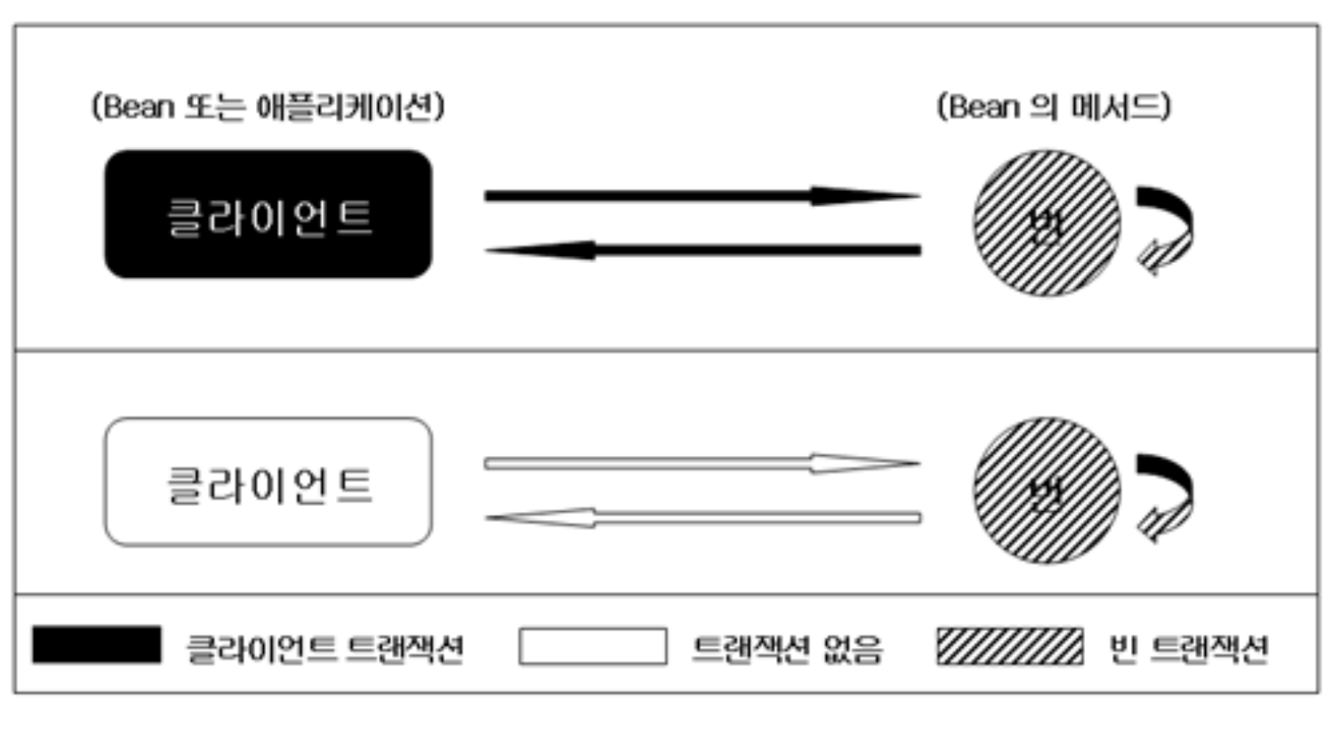
Transaction Propagation(Required)



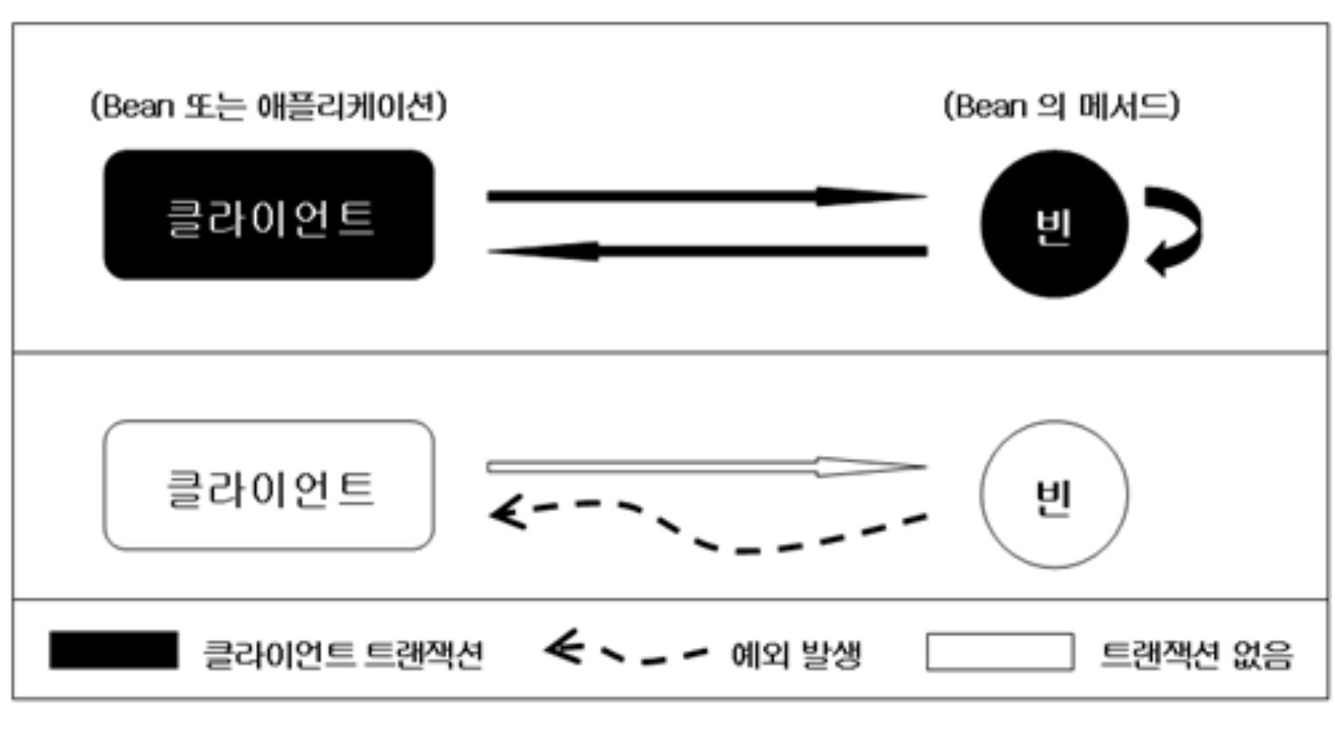
Transaction Propagation(Supports)



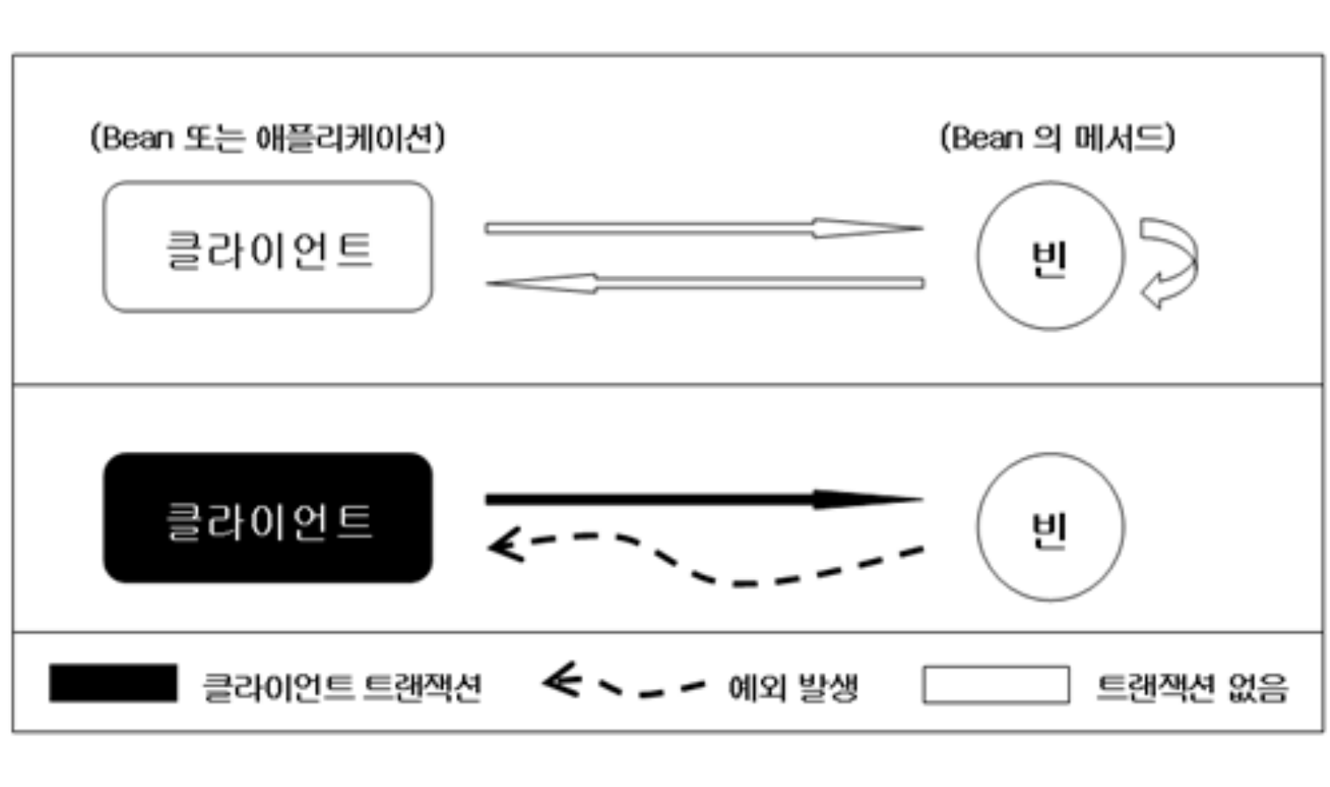
Transaction Propagation(Requires New)



Transaction Propagation(Mandatory)



Transaction Propagation(Naver)



Isolation 속성과 트랜젝션 격리레벨

속성 값	설명
DEFAULT	기본 설정을 사용한다.
READ_UNCOMMITTED	다른 트랜잭션에서 Commit하지 않은 데이터를 읽을 수 있다.
READ_COMMITTED	다른 트랜잭션에서 Commit한 데이터를 읽을 수 있다.
REPEATABLE_READ	처음에 읽은 데이터와 두 번째 읽은 데이터가 동일한 값을 갖는다.
SERIALIZABLE	동일한 데이터에 대해서 동시에 두 개 이상의 트랜잭션이 수행될 수 없다.

Annotation을 이용한 트랜잭션 처리

Annotation 관련 설정

Annotation 기반의 트랜잭션을 위해서 다음 설정이 필요함

```
<tx:annotation-driven transaction-manager="txManager"/>
```

TransactionManager를 무엇을 사용할지는 설정파일에 등록된 bean을 사용함.

@Transactional 사용

각 클래스와 메서드에 지정할 수 있음.
메서드가 클래스보다 더 우선함.

```
@Transactional(readOnly = true)
public class UserServiceImpl implements UserService {
public UserVO getUser(UserVO vo) {
...
}

@Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW)
public void updateFoo(Foo foo) {
...
}
}
```

@Transactional 의 속성

속성	필수여부	기본값	설명
Propagation	선택	REQUIRED	트랜잭션 전파 방법을 설명한다.
Isolation	선택	DEFAULT	트랜잭션 격리 레벨. 데이터베이스가 지원하는 기본 값으로 설정된다.
Timeout	선택	-1	트랜잭션 타임아웃 값 (초단위 설정)
readOnly	선택	false	트랜잭션이 읽기만 가능한가?
rollbackFor	선택		트랜잭션을 Rollback할 예외 타입을 지정한다. 콤마로 구분하여 여러 개를 나열 할 수 있다. 예를 들어 com.MyException, com.AnotherException 등으로 설정한다.
noRollbackFor	선택		트랜잭션을 Rollback하지 않을 예외 타입을 지정한다.