목차

1. DB 접속 정보 설정

DB 접속정보를 설정하기 위해서는 context-datasouce.xml 에 DB 접속정보를 설정해 주면 됩니다.
DBMS 종류에 따라 드라이버는 다르게 설정해야 함
MySql 을 사용하는 경우 : com.mysql.jdbc.Driver 드라이버를 사용
Oracle 을 사용하는 경우 : oracle.jdbc.driver.OracleDriver 드라이버를 사용

- servlet.xml 에서

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:beans="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"

    xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd
        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.xsd">


    <!-- 스프링의 DispatcherServet 에게 정적인 자원을 알려준다 -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views -->
    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>


    <!-- step1. annotation 설정 -->
    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />

    <!-- step2. component scan 설정 -->
    <context:component-scan base-package="aop01" />

    <!-- step3. 데이터베이스 설정 -->
    <!-- 커넥션 설정 -->
    <beans:bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">
        <beans:property name="driverClassName"
value="oracle.jdbc.driver.OracleDriver" />
        <beans:property name="url"
```

```
value="jdbc:oracle:thin:@//localhost:1521/pdborcl" />
        <beans:property name="username"                    value="tester1" />
        <beans:property name="password"                    value="1234" />

        <beans:property name="defaultAutoCommit"           value="true"/>
        <beans:property name="poolPreparedStatements"      value="true"/>
        <beans:property name="cacheState"                  value="true"/>
    </beans:bean>


    <!-- SessionFactory 설정 :: MyBatis 가 사용할 Database 에 연결하도록 설정 -->
    <beans:bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <beans:property name="dataSource"          ref="dataSource" />
        <!-- <beans:property name="configLocation"     value="classpath:Configuration.xml"
/>  -->

        <!-- mybatis 디렉토리에 xml 파일만 추가해주면 알아서 xml 내의 쿼리 자동 인식 -->
        <!-- <beans:property name="mapperLocations"
value="classpath*:mybatis/**/mapper*.xml" />  -->
    </beans:bean>

    <!-- MyBatis 의 CRUD 템플릿을 사용할 수 있도록 설정 -->
    <beans:bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate"
scope="singleton">
        <beans:constructor-arg index="0" ref="sqlSessionFactory" />
    </beans:bean>


</beans:beans>
```

2. 트랜잭션 설정

트랜잭션을 설정하기 위해서 xml 기반의 설정과 annotation 기반의 설정 모두 가능합니다. Spring Transaction 적용 방법은 크게 2 가지 방법입니다.

1. annotation 기반의 설정 - @Transactional 사용
2. AOP 기반의 설정 - <tx:advice> 사용

- AOP 기반의 설정은 proxy 에 의해 일어나기 때문에 cglib 라이브러리와 인터페이스가 있는 @Service 클래스에서 실행되어야 오류가 없습니다.

2.1 servlet-context.xml 설정

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

    xsi:schemaLocation="http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        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.xsd

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


    <!-- 스프링의 DispatcherServet 에게 정적인 자원을 알려준다 -->
    <resources mapping="/resources/**" location="/resources/" />

    <!-- Resolves views -->
    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
    </beans:bean>
```

```
    <!-- step1. annotation 설정 -->
    <!-- Enables the Spring MVC @Controller programming model -->
    <annotation-driven />

    <!-- step2. component scan 설정 -->
    <context:component-scan base-package="aop01" />

    <!-- step3. 데이터베이스 설정 -->
    <!-- 커넥션 설정 -->
    <beans:bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">
        <beans:property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
        <beans:property name="url" value="jdbc:oracle:thin:@//localhost:1521/pdborcl" />
        <beans:property name="username"   value="tester1" />
        <beans:property name="password"   value="1234" />

        <beans:property name="defaultAutoCommit"          value="true"/>
        <beans:property name="poolPreparedStatements"     value="true"/>
        <beans:property name="cacheState"                 value="true"/>
    </beans:bean>


    <!-- SessionFactory 설정 :: MyBatis가 사용할 Database에 연결하도록 설정 -->
    <beans:bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <beans:property name="dataSource"  ref="dataSource" />
        <!-- <beans:property name="configLocation"  value="classpath:Configuration.xml" />
-->

        <!-- mybatis 디렉토리에 xml 파일만 추가해주면 알아서 xml 내의 쿼리 자동 인식 -->
        <!-- <beans:property name="mapperLocations"
value="classpath*:mybatis/**/mapper*.xml" />  -->
    </beans:bean>

    <!-- MyBatis의 CRUD 템플릿을 사용할 수 있도록 설정 -->
    <beans:bean id="sqlSession"
                class="org.mybatis.spring.SqlSessionTemplate"
                scope="singleton">
        <beans:constructor-arg index="0" ref="sqlSessionFactory" />
    </beans:bean>


    <!-- step4. 트랜잭션 설정 -->
    <beans:bean id="txManager"
                class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <beans:property name="dataSource" ref="dataSource" />
    </beans:bean>
```

```
    <aop:config proxy-target-class="true">
        <aop:pointcut id="serviceOperation"
                      expression="execution(public * *..*Service*.*(..))" />
        <aop:advisor id="transactionAdvisor"
                     advice-ref="txAdvice" pointcut-ref="serviceOperation" />
    </aop:config>

    <tx:advice id="txAdvice" transaction-manager="txManager">
        <tx:attributes>
            <tx:method name="get*"     read-only="true" />
            <tx:method name="*insert*" rollback-for="Exception" propagation="REQUIRED" />
            <tx:method name="*update*" rollback-for="Exception" propagation="REQUIRED" />
            <tx:method name="delete*"  rollback-for="Exception" propagation="REQUIRED" />
            <tx:method name="trans*"   rollback-for="Exception" propagation="REQUIRED" />
        </tx:attributes>
    </tx:advice>


    <!-- step5. mybatis 연동 설정 : base-package 에 매퍼 인터페이스 패키지를 지정 -->
    <mybatis:scan base-package="com.lecture.spring.mybatis" />
    <beans:bean id="sqlSessionFactory"
                class="org.mybatis.spring.SqlSessionFactoryBean">
        <beans:property name="dataSource" ref="dataSource" />

        <!-- mybatis 설정을 위한 설정 파일 경로를 지정하시면 됩니다 -->
        <beans:property name="configLocation" value="classpath:Configuration.xml" />
    </beans:bean>

</beans:beans>
```

- 참고사항

속성	설 명	사 용 예
isolation	Transaction 의 isolation Level 정의하는 요소. 별도로 정의하지 않으면 DB 의 Isolation Level 을 따름.	@Transactional(isolation=Isolation.DEFAULT)
noRollbackFor	정의된 Exception 목록에 대해서는 rollback 을 수행하지 않음.	@Transactional(noRollbackFor=NoRoleBackTx.class)
noRollbackForClassName	Class 객체가 아닌 문자열을 이용하여 rollback 을 수행하지 않아야 할	@Transactional(noRollbackForClassName="NoRoleBackT

속성	설 명	사 용 예
	Exception 목록 정의	x")
propagation	Transaction 의 propagation 유형을 정의하기 위한 요소	@Transactional(propagation =Propagation.REQUIRED)
readOnly	해당 Transaction 을 읽기 전용 모드로 처리 (Default = false)	@Transactional(readOnly = true)
rollbackFor	정의된 Exception 목록에 대해서는 rollback 수행	@Transactional(rollbackFor= RoleBackTx.class)
rollbackForClassName	Class 객체가 아닌 문자열을 이용하여 rollback 을 수행해야 할 Exception 목록 정의	@Transactional(rollbackForClassName="RoleBackTx")
timeout	지정한 시간 내에 해당 메소드 수행이 완료되지 않은 경우 rollback 수행. -1 일 경우 no timeout (Default = -1)	@Transactional(timeout=10)

2.2 DAO 구현

```
@Repository
public class UserDAOImpl implements UserDAO{
//이 구문을 추가하여야 application-Context.xml 에서 선언한 sqlSession 을 받아 처리 가능.

@Autowired
@Qualifier("sqlSession")
private SqlSession sqlsession;

@Override
public void UserInfo(springdemo.UserInfo userInfo) throws Exception{
sqlsession.insert("org.mybatis.user.insertUser", userInfo);
}
```

2.3 Service 구현

```
@Service
public class ServiceBoard implements IServiceBoard  {

   @Autowired
```

```
    private MapperBoard boardMapper;
    …

    @Override
    public void transTest(ModelBoard board) throws SQLException {

        boardMapper.updateBoard(board);
        boardMapper.updateBoard(null);

        throw new SQLException("make an exception intentionally");
    }
}
```

2.4 실행 파일

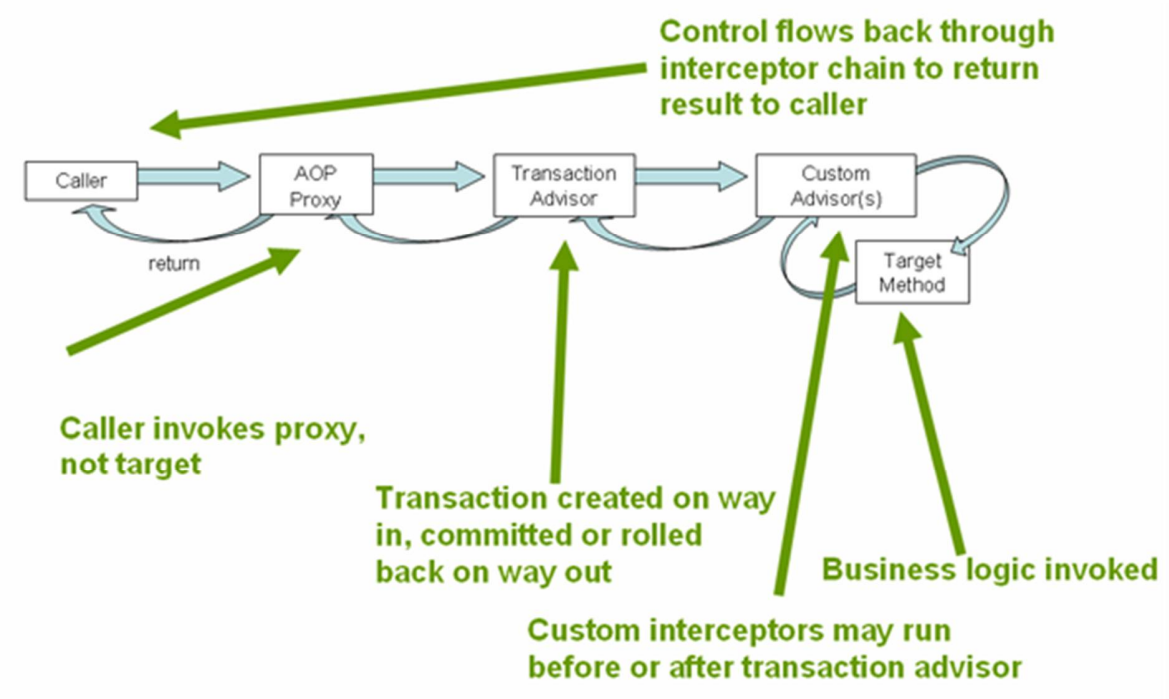
```
public class testServiceBoard {

    // SLF4J Logging
    private static Logger logger = LoggerFactory.getLogger(testServiceBoard.class);

    private IServiceBoard boardService ;

    @Before
    public void setUp() throws Exception {
        try {

            ApplicationContext ctx = new
ClassPathXmlApplicationContext("file:src/main/webapp/WEB-INF/spring/appServlet/servlet-
context.xml");
            boardService = ctx.getBean(IServiceBoard.class);
        } catch (BeansException e) {
            logger.error("testAnnotation", e );
        }
    }

    @Test
    public void testTransTest() {
        try {

            String boardnm = new Date().toString();
```

```
            ModelBoard board = new ModelBoard();
            board.setBoardcd("test");
            board.setBoardnm( boardnm );
            board.setUseYN(true);

            boardService.transTest(board);
            assertTrue(false);
        } catch (Exception e) {
            //logger.error("testAnnotation", e );
            assertTrue(true);
        }

    }
}
```

위에 선언한 것 처럼 transTest() 실행 시 오류 발생으로 모두 rollback 이 된다.

2.5 실행 로그

```
o.s.j.d.DataSourceTransactionManager - Using transaction object [org.springframework.jdbc.datasource.DataSour
o.s.j.d.DataSourceTransactionManager - Creating new transaction with name [com.lecture.spring.service.Servicel
o.s.j.d.DataSourceTransactionManager - Acquired Connection [jdbc:mysql://localhost:3306/springboard, UserName=
o.s.j.d.DataSourceTransactionManager - Switching JDBC Connection [jdbc:mysql://localhost:3306/springboard, Us
o.s.t.s.TransactionSynchronizationManager - Bound value [org.springframework.jdbc.datasource.ConnectionHolder
o.s.t.s.TransactionSynchronizationManager - Initializing transaction synchronization
o.s.t.i.TransactionInterceptor - Getting transaction for [com.lecture.spring.service.ServiceBoard.transTest]
org.mybatis.spring.SqlSessionUtils - Creating a new SqlSession
org.mybatis.spring.SqlSessionUtils - Registering transaction synchronization for SqlSession [org.apache.ibati
o.s.t.s.TransactionSynchronizationManager - Bound value [org.mybatis.spring.SqlSessionHolder@7a0f244f] for ke
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.springframework.jdbc.datasource.ConnectionHo
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.springframework.jdbc.datasource.ConnectionHo
o.m.s.t.SpringManagedTransaction - JDBC Connection [jdbc:mysql://localhost:3306/springboard, UserName=user01@
c.l.s.m.MapperBoard.updateBoard - ==>  Preparing: UPDATE TB_Board SET boardnm = ? , UseYN = ? WHERE boardcd =
c.l.s.m.MapperBoard.updateBoard - ==> Parameters: Sun Aug 02 10:12:17 KST 2015(String), true(Boolean), test(S
c.l.s.m.MapperBoard.updateBoard - <==    Updates: 1
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.mybatis.spring.SqlSessionHolder@7a0f244f] fo
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.mybatis.spring.SqlSessionHolder@7a0f244f] fo
org.mybatis.spring.SqlSessionUtils - Releasing transactional SqlSession [org.apache.ibatis.session.defaults.D
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.mybatis.spring.SqlSessionHolder@7a0f244f] fo
org.mybatis.spring.SqlSessionUtils - Fetched SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession
c.l.s.m.MapperBoard.updateBoard - ==>  Preparing: UPDATE TB_Board SET boardnm = ? , UseYN = ? WHERE boardcd =
c.l.s.m.MapperBoard.updateBoard - ==> Parameters: null, null, null
c.l.s.m.MapperBoard.updateBoard - <==    Updates: 0
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.mybatis.spring.SqlSessionHolder@7a0f244f] fo
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.mybatis.spring.SqlSessionHolder@7a0f244f] fo
org.mybatis.spring.SqlSessionUtils - Releasing transactional SqlSession [org.apache.ibatis.session.defaults.D
o.s.t.i.TransactionInterceptor - Completing transaction for [com.lecture.spring.service.ServiceBoard.transTes
o.s.t.i.RuleBasedTransactionAttribute - Applying rules to determine whether transaction should rollback on ja
o.s.t.i.RuleBasedTransactionAttribute - Winning rollback rule is: RollbackRuleAttribute with pattern [Exceptio
o.s.j.d.DataSourceTransactionManager - Triggering beforeCompletion synchronization
org.mybatis.spring.SqlSessionUtils - Transaction synchronization deregistering SqlSession [org.apache.ibatis.
o.s.t.s.TransactionSynchronizationManager - Removed value [org.mybatis.spring.SqlSessionHolder@7a0f244f] for
org.mybatis.spring.SqlSessionUtils - Transaction synchronization closing SqlSession [org.apache.ibatis.sessio
o.s.t.s.TransactionSynchronizationManager - Retrieved value [org.springframework.jdbc.datasource.ConnectionHo
o.s.j.d.DataSourceTransactionManager - Initiating transaction rollback
o.s.j.d.DataSourceTransactionManager - Rolling back JDBC transaction on Connection [jdbc:mysql://localhost:330
o.s.j.d.DataSourceTransactionManager - Triggering afterCompletion synchronization
o.s.t.s.TransactionSynchronizationManager - Clearing transaction synchronization
o.s.t.s.TransactionSynchronizationManager - Removed value [org.springframework.jdbc.datasource.ConnectionHold
o.s.j.d.DataSourceTransactionManager - Releasing JDBC Connection [jdbc:mysql://localhost:3306/springboard, Us
o.s.jdbc.datasource.DataSourceUtils - Returning JDBC Connection to DataSource
com.lecture.spring.testMain - testAnnotation
```



http://docs.spring.io/spring/docs/current/spring-framework-reference/html/transaction.html

3. Transactional Annotation

@Transactional Annotation 을 이용한 처리

```
@Service
public class TransactionSample {

   @Autowired(required = true)
   private BoardDao boardDao;

   @Transactional
   public int updateBoard(BoardModel[] modelArray) throws Exception {
      int result = 0;

      for (BoardModel ele: modelArray) {
         result += boardDao.updateBoard(ele);
      }

      return result;
   }
}
```

위와 같이 메서드 선언부에 @Transactional 어노테이션을 명시하면 호출스택에
TransactionInterceptor 가 추가되는데,
서비스 레벨에서 발생한 예외를 TransactionInterceptor 가 받을 수 있도록만 작성하면 rollback 은
자동으로 이뤄진다.

DataSourceTransactionManager 를 이용한 처리

메서드 본문에서 메서드 본문 전체가 아닌 본문 중 특정 지역별로 트랜잭션을 묶어야 할 때 사용한다.

```
@Service
public class TransactionSample {

   @Autowired(required = true)
   private DataSourceTransactionManager transactionManager;

   @Autowired(required = true)
   private BoardDao boardDao;
```

```
    public int updateBoard(BoardModel[] modelArray) {
        DefaultTransactionDefinition def = new DefaultTransactionDefinition();
        def.setPropagationBehavior(TransactionDefinition.PROPAGATION_REQUIRED);
        TransactionStatus status = transactionManager.getTransaction(def);

        int result = 0;

        try {
            for (BoardModel ele: modelArray) {
                result += boardDao.updateBoard(ele);
            }
            transactionManager.commit(status);
        } catch (Exception e) {
            transactionManager.rollback(status);
            throw e;
        }

        return result;
    }
}
```

10 ~ 12, 21, 24 번 줄과 같은 방식으로 commit 과 rollback 을 제어할 수 있다.

4. MyBatis 연동 설정

- mybatis 는 스프링과 쉽게 연동하여 sql 처리를 수월하게 해줄 수 있는 sql mapper 입니다.

- servlet.xml 에서

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


    xsi:schemaLocation="http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        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.xsd

        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd
        http://mybatis.org/schema/mybatis-spring
        http://mybatis.org/schema/mybatis-spring.xsd">

    // 이어서

    <!-- step5. mybatis 연동 설정 : base-package 에 매퍼 인터페이스 패키지를 지정 -->
    <mybatis:scan base-package="com.lecture.spring.mybatis" />

</ beans:beans>
```

5. Reference

http://www.jayway.com/2013/05/12/getting-started-with-gradle/
http://www.slipp.net/wiki/pages/viewpage.action?pageId=12878060

https://www.credera.com/blog/custom-application-development/converting-spring-boot-project-maven-gradle-sts/
http://stackoverflow.com/questions/13925724/providedcompile-without-war-plugin

http://docs.spring.io/spring/docs/current/spring-framework-reference/html/transaction.html

http://hellogk.tistory.com/94
http://hellowk1.blogspot.kr/2014/02/spring-framework-transaction-aop.html
http://hellowk1.blogspot.kr/2015/03/spring-framework-transaction-with.html

http://barunmo.blogspot.kr/2013/06/mybatis.html

http://egloos.zum.com/springmvc/v/499291
http://blog.outsider.ne.kr/870

http://noritersand.tistory.com/198
http://spring.io/guides/gs/managing-transactions/
http://wiki.gurubee.net/pages/viewpage.action?pageId=26741432
http://docs.spring.io/spring/docs/current/spring-framework-reference/html/transaction.html
http://blog.outsider.ne.kr/870