Spring db

라이브러리 추가 (spring-jdbc, dbcp)

pom.xml - spring-jdbc, dbcp

```
<dependency>
    <groupId>commons-dbcp</groupId>
    <artifactId>commons-dbcp</artifactId>
    <version>X.x</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jdbc</artifactId>
    <version>X.x.x.RELEASE</version>
</dependency>
```

라이브러리 추가(myBatis)

pom.xml - mybatis-spring, mybatis

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis-spring</artifactId>
    <version>X.x.x</version>
</dependency>
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>X.x.x</version>
</dependency>
```

라이브러리 추가 **(oracle)**

pom.xml - ojdbc.jar 파일을 클래스 패스 설정

```
<repositories>
    <repository>
      <id>oracle</id>
      <name>ORACLE JDBC Repository</name>
      <url>http://maven.jahia.org/maven2</url>
    </repository>
</repositories>

<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.1.0.7.0</version>
</dependency>
```

데이터베이스 정보 설정 – application-context

✓ DriverManager를 이용한 DataSource 설정

: 일반 JDBC 방식

```
<bean id="dataSource"
      class="org.springframework.jdbc.datasource.
                                    DriverManagerDataSource"
      p:driverClassName="oracle.jdbc.driver.OracleDriver"
      p:url="jdbc:oracle:thin:@localhost:1521:XE"
      p:username="hr"
      p:password="hr" />
```

datasource 설정 – application-context

✓ 커넥션 풀을 이용한 DataSource 설정

: DBCP(Jakara Commons Database Connection Pool) API를 이용한 설정

```
<bean id="dataSource"
      class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close"
      p:driverClassName="oracle.jdbc.driver.OracleDriver"
      p:url="jdbc:oracle:thin:@localhost:1521:XE"
      p:username="hr"
      p:password="hr" />
```

spring + mybatis 연동 – xml : – application-con

SqlSession을 구현한 SqlSessionTemplate 객체 활용

```
<bean id="sqlSessionFactory"
      class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="configLocation"
   value="classpath:config/mybatis/sqlMapConfig.xml" />
  <property name="mapperLocations"
       value="classpath*:config/sqlmap/oracle/*.xml" />
</bean>
<bean id="sqlSessionTemplate"
      class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg ref="sqlSessionFactory" />
</bean>
```

myBatis 설정파일(sqlMapConfig.xml)

```
<configuration>
   <typeAliases>
      <typeAlias
            alias="member" type="member.vo.MemberVO" />
   </typeAliases>
</configuration>
```

Java + myBatis 연동

```
@Repository
public class MemberDAO {
  @Autowired
  private SqlSessionTemplate sqlSessionTemplate;
  public List<MemberVO> selectMember() throws Exception
  {
    List<MemberVO> list =
              sqlSessionTemplate.selectList(
                   "member.dao.Member.getMemberList");
    return list;
  }
}
```

트랜잭션

- ✓ Container 가 제공되는 가장 대표적 서비스
- ✓ 설정파일내 TransactionManager 설정만으로 소스코드 내에 Transaction 관련 코드를 사용하지 않고 자동 Transaction 이 가능
- ✓ Spring 에서는 서로 다른 트랜잭션 API를 지원하며 지원 종류는 JDBC, JTA, JDO, JPA, 하이버네이트 등이 있음
- ✓ 트랜잭션 사용방법은 환경설정파일인 xml 에 선언하여 사용하는 방법과 프로그램에서 직접 제어하는 방법 2가지가 있음

AOP 방식의 트랜잭션

1. <beans> 태그에 xmlns:tx 관련 부분을 추가

```
<beans xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation=
      "http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd">
```

2. 트랜잭션 매니져 설정 : 실제 트랜잭션 관리 처리

```
<bean id="transactionManager"
      class="org.springframework.jdbc.datasource.
                                    DataSourceTransactionManager"
      p:dataSource-ref="dataSource" />
```

AOP 방식의 트랜잭션

3. 트랜잭션 매니져를 어드바이스로 설정

```
<tx:advice id="txAdvice"
           transaction-manager="transactionManager">
  <tx:attributes>
    <tx:method name="*" rollback-for="Exception" />
  </tx:attributes>
</tx:advice>
```

AOP 방식의 트랜잭션

4. 트랜잭션 AOP 설정을 통한 적용

```
<aop:config>
    <aop:pointcut id="tranMethod"
    expression=
            "execution(public * member.*ServiceImpl.*(..)) "/>
    <aop:advisor advice-ref="txAdvice"
                 pointcut-ref="tranMethod" />
</aop:config>
```

어노테이션 방식의 트랜잭션

1. annotation-driven 설정

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

2. 클래스에 어노테이션 표기법 추가

```
@Service("memberService")
public class MemberServiceImpl implements MemberService {
 …… 생략
 @Transactional(rollbackFor=Exception.class)
 public void registMember(MemberVO memberVO) throws Exception {
   memberDAO.insertMember(memberVO);
 }
}
```