Chapter 6

6.1.테이블 생성 및 개발

6.1.1. 데이터베이스의 테이블 생성

```
CREATE TABLE tbl_member (
    userid varchar(50) not null
    , userpw varchar(50) not null
    , username varchar(50) not null
    , email varchar(100)
    , regdate timestamp default now()
    , updatedate timestamp default now()
    , primary key(userid)
);
```

6.1.2. 도메인 객체를 위한 클래스 설계

```
public class MemberVO {
    private String userid;
    private String userpw;
    private String username;
    private String email;
    private Date regdate;
    private Date updatedate;
  // alt + shift + s, o -- toString()
  // alt + shift + s, r -- getter/setter()
}
```

6.2 DAO 인터페이스의 작성

6.1.3. MemberDAO 인터페이스 생성

```
public interface MemberDAO {

    public String getTime();

    public void insertMember(MemberVO vo);

}
```

6.3. XML Mapper의 작성

- XML로 작성된 Mapper의 위치(저장 경로) 결정
- XML Mapper 파일을 작성하고 필요한 DTD 추가
- SQL 작성

6.3.1. Mapper 파일의 저장 경로

```
src/main/resources/mappers/memberMapper.xml
```

6.3.2. memberMapper.xml의 작성

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace ="com.spring.web.MemberMapper">
    <select id="getTime" resultType="string">
        SELECT now()
    </select>

    <insert id="insertMember">
        INSERT INTO tbl_member (
                userid
                , userpw
                , username
                , email
        )
        VALUES (
                #{userid}
                , #{userpw}
                , #{username}
                , #{email}
        )
    </insert>
</mapper>
```

6.3.3. myBatis-Spring에서 XML mapper 인식

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"></property>
  <property name="configLocation" value="classpath:/mybatis-config.xml"></property>
  <property name="mapperLocations" value="classpath:/mappers/**/*Mapper.xml"></property>
</bean>
```

6.2. DAO 인터페이스 구현

6.2.1. SqlSessionTemplate의 설정

```
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
        <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory"></constructor-arg>
</bean>
```

6.2.2. MemberDAOImpl 작성하기(구현 클래스)

- @Repository: DAO를 스프링에 인식시키기 위한 어노테이션

```
@Repository
public class MemberDAOImpl implements MemberDAO{

    @Inject
    private SqlSession sqlSession;

    private static final String namespace = "com.spring.web.MemberMapper";

    @Override
    public String getTime() {
        return sqlSession.selectOne(namespace + ".getTime");
    }

    @Override
    public void insertMember(MemberVO vo) {
        sqlSession.insert(namespace + ".insertMember", vo);
    }

    @Override
    public MemberVO readMember(String userid) throws Exception {
        return (MemberVO) sqlSession.selectOne(namespace + ".selectMember", userid);
    }

    @Override
    public MemberVO readWithePW(String userid, String userpw) throws Exception {
        Map<String, Object> paramMap = new HashMap<String, Object>();

        paramMap.put("userid", userid);
        paramMap.put("userpw", userpw);
        return sqlSession.selectOne(namespace + ".readWithPW", paramMap);
    }
}
```

6.4. 스프링에 빈으로 등록하기(root-context.xml)

```
<context:component-scan base-package="com.spring.persistence"></context:component-scan>
```

6.5. 테스트 코드의 작성

- src/test/java/{패키지}/MemberDAOTest

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"file:src/main/webapp/WEB-INF/spring/**/root-context.xml"})
public class MemberDAOTest {
```

```
    @Inject
    private MemberDAO dao;

    @Test
    public void testTime() {
        System.out.println(dao.getTime());
    }

    @Test
    public void testInsertMember() {
        MemberVO vo = new MemberVO();
        vo.setUserid("user08");
        vo.setUserpw("user08");
        vo.setUsername("USER08");
        vo.setEmail("user08@aaa.com");

        dao.insertMember(vo);
    }
}
```

6.6. MyBatis의 로그 log4jdbc-log4j2

6.6.1. pom.xml에 dependancy 추가

```
<!-- https://mvnrepository.com/artifact/org.bgee.log4jdbc-log4j2/log4jdbc-log4j2-jdbc4 -->
<dependency>
    <groupId>org.bgee.log4jdbc-log4j2</groupId>
    <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
    <version>1.16</version>
</dependency>
```

6.6.2. root-context에 log4jdbc 적용

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value ="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
    <property name="url" value="jdbc:log4jdbc:mysql:/{ip}:3306/{db}?useSSL=false"></property>
    <property name="username" value="{dbid}"></property>
    <property name="password" value="{dbpw}"></property>
</bean>
```

6.6.3. log4jdbc-log4j2 동작을 위한 설정파일 추가

- log4jdbc.log4j2.properties 파일 작성

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```

- logback.xml 작성

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <include resource="org/springframework/boot/logging/logback/base.xml"/>

    <!-- log4jdbc-log4j2 -->
    <logger name="jdbc.sqlonly"             level="DEBUG"/>
    <logger name="jdbc.sqltiming"           level="INFO"/>
    <logger name="jdbc.audit"               level="WARN"/>
    <logger name="jdbc.resultest"           level="ERROR"/>
    <logger name="jdbc.resultsettalbe"      level="ERROR"/>
    <logger name="jdbc.connection"          level="INFO"/>

</configuration>
```

6.7 Mybatis의 #{} 문법

6.7.1. MemberDAO 인터페이스 작성

```
public interface MemberDAO {

    public String getTime();

    public void insertMember(MemberVO vo);

    public MemberVO readMember(String userid) throws Exception;

    public MemberVO readWithePW(String userid, String userpw) throws Exception;
}
```

6.7.2. resources/mappers/memberMapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace ="com.spring.web.MemberMapper">
    <select id="getTime" resultType="string">
        SELECT now()
    </select>

    <insert id="insertMember">
        INSERT INTO tbl_member (
                userid
                , userpw
```

```
                , username
                , email
        )
        VALUES (
                #{userid}
                , #{userpw}
                , #{username}
                , #{email}
        )
    </insert>
    <select id="selectMember" resultType="com.spring.domain.MemberVO">
        SELECT
            *
        FROM tbl_member
        WHERE userid = #{userid}
    </select>
    <select id="readWithPW" resultType="com.spring.domain.MemberVO">
        SELECT
            *
        FROM tbl_member
        WHERE userid = #{userid}
        AND userpw = #{userpw}
    </select>
</mapper>
```

6.7.3. org.zerock.persistence.MemberDAOImpl 구현 클래스 작성

- readWithPW()의 경우 파라미터가 2개 이상 전달되는 경우, Map 타입의 객체를 구성해서 파라미터로 사용

```
@Repository
public class MemberDAOImpl implements MemberDAO{

    @Inject
    private SqlSession sqlSession;

    private static final String namespace = "com.spring.web.MemberMapper";

    @Override
    public String getTime() {
        return sqlSession.selectOne(namespace + ".getTime");
    }

    @Override
    public void insertMember(MemberVO vo) {
        sqlSession.insert(namespace + ".insertMember", vo);
    }

    @Override
    public MemberVO readMember(String userid) throws Exception {
        return (MemberVO) sqlSession.selectOne(namespace + ".selectMember", userid);

    }
```

```
    @Override
    public MemberVO readWithePW(String userid, String userpw) throws Exception {
        Map<String, Object> paramMap = new HashMap<String, Object>();

        paramMap.put("userid", userid);
        paramMap.put("userpw", userpw);
        return sqlSession.selectOne(namespace + ".readWithPW", paramMap);
    }
}
```