전자정부 표준프레임워크
실행환경(데이터처리)
실습

eGovFrame

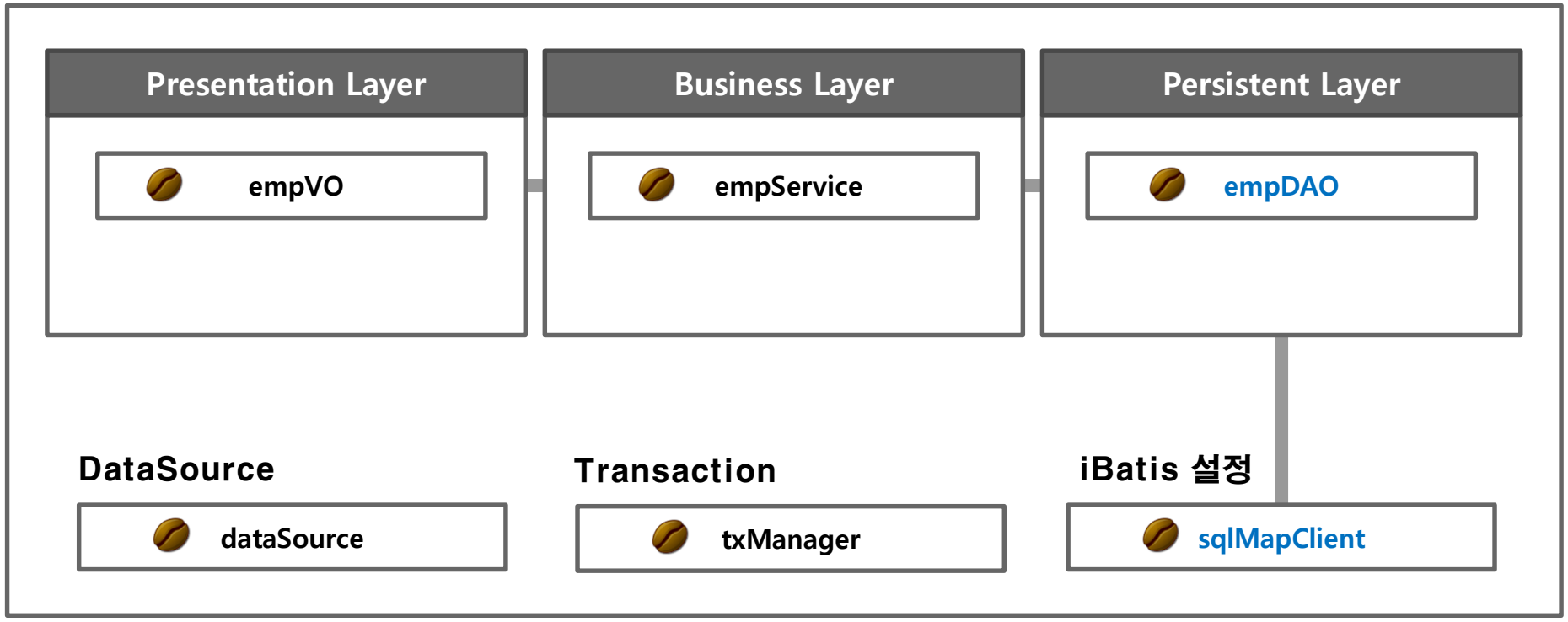
Contents

실습 개요

- 실습을 통해 iBatis와 MyBatis 설정 및 사용법에 대해 알아본다.
- 실습 순서
  - iBatis 예제
    1. 구동 환경 설정 (DataSource 빈 설정, 스프링 연동 빈 설정, iBatis XML 설정)
    2. Service, DAO 클래스 작성
    3. 오브젝트-SQL 매핑 파일 작성
    4. CRUD 테스트
  - MyBatis 예제
    1. 구동 환경 설정 (DataSource 빈 설정, 스프링 연동 빈 설정, MyBatis XML 설정)
    2. Service, DAO 클래스 작성
    3. 오브젝트-SQL 매핑 파일 작성
    4. CRUD 테스트

LAB 204-iBatis 실습

❑ lab204-ibatis 프로젝트의 beans



Step 1. 구동 환경 설정

❑ 1. Hsqldb 초기화 스크립트

- /lab204-ibatis/src/test/resources/META-INF/testdata/sample_schema_hsql.sql 를 확인한다.
- 현 실습 프로젝트에서는 편의상 매 테스트 케이스 재실행 시 관련 table 을 drop/create 하고 있음.

❑ 2. dataSource 설정

- <bean id="dataSource" .../>를 이용한 방법
- <jdbc:embedded-database .../>를 이용한 방법

Step 1. 구동 환경 설정

❑ 2-1. dataSource 설정 (1/2)

- /lab204-ibatis/src/test/resources/META-INF/spring/context-datasource.xml 에 dataSource 빈 설정을 추가한다.

```
<!-- TODO [Step 1-2] dataSource 설정 -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
<property name="driverClassName" value="${db.driver}" />
<property name="url" value="${db.dburl}" />
<property name="username" value="${db.username}" />
<property name="password" value="${db.password}" />
<property name="defaultAutoCommit" value="false" />
<property name="poolPreparedStatements" value="true" />
</bean>
```

- /lab204-ibatis/src/test/resources/META-INF/spring/context-common.xml 에 context:property-placeholder 설정을 추가한다.

```
<!-- TODO [Step 1-2] PropertyPlaceholderConfigurer 설정 -->
<context:property-placeholder location="classpath:/META-INF/spring/jdbc.properties" />
```

Step 1. 구동 환경 설정

❑ 2-1. dataSource 설정 (2/2)

- /lab204-ibatis/src/test/resources/META-INF/spring/jdbc.properties 를 확인한다.

```
#TODO [Step 1-2] dataSource 설정
db.driver=org.hsqldb.jdbcDriver
db.dburl=jdbc:hsqldb:mem:testdb
#db.dburl=jdbc:hsqldb:hsql://localhost/sampledb
db.username=sa
db.password=

#TODO [Step 3-1] Sql 로깅 설정을 위한 dburl 변경
#db.driver=net.sf.log4jdbc.DriverSpy
#db.dburl=jdbc:log4jdbc:hsqldb:mem:testdb
#db.dburl=jdbc:log4jdbc:hsqldb:hsql://localhost/sampledb
#db.username=sa
#db.password=
```

cf.) 위에서 기본으로 memory DB 형식으로 자동 구동하도록 되어 있으나 dburl=jdbc:hsqldb:hsql://localhost/sampledb 과 같이 변경 시에는 /lab204-ibatis/db 상에서 (외부 탐색기에서) runHsqlDB.cmd 를 실행하여 DB Server 를 구동하고 테스트할 수도 있다.

Step 1. 구동 환경 설정

❑ 3. transaction 설정

- /lab204-ibatis/src/test/resources/META-INF/spring/context-transaction.xml 를 작성한다.

```
<!-- TODO [Step 1-3] transaction 설정 -->
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

cf.) 여기서는 transaction manager 만을 설정하였고, TestCase 내에서 전역 @Transactional 설정으로 트랜잭션을 일괄 지정하고 있으나, 보통 AOP 형식(tx:aop)의 트랜잭션 대상 지정으로 비즈니스 서비스의 메서드에 일괄 지정하는 경우가 많다.

cf2.) @Transactional Annotation 으로 대상 메서드에 개별로 따로 지정할 수도 있다.

❑ 4. Spring 의 iBATIS 연동 설정

- /lab204-ibatis/src/test/resources/META-INF/spring/context-sqlMap.xml 를 작성한다.

```
<!-- TODO [Step 1-4] Spring 의 iBATIS 연동 설정 -->
<bean id="sqlMapClient"
class="egovframework.rte.psl.orm.ibatis.SqlMapClientFactoryBean">
<property name="configLocation"
value="classpath:/META-INF/sqlmap/sql-map-config.xml" />
<!--
<property name="mappingLocations"
value="classpath:/META-INF/sqlmap/mappings/lab-*.xml" />
 -->
<property name="dataSource" ref="dataSource" />
</bean>
```

- 최신 프레임워크 환경에서는 sql-map-config.xml 내에 개별 sql 맵핑 파일일 일일이 지정하는 것이 아니라 위의 mappingLocations 영역을 주석 해제하여 Spring 의 ResourceLoader 형식으로 패턴 매칭에 의거한 일괄 로딩으로 처리가 가능하다. (단, 테스트 결과 CacheModel 등의 일부 기능에서 문제가 발생하는 경우가 있으므로 사용에 유의할것!!)

❑ 5. iBATIS 의 sql-map-config 설정 파일 작성

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/sql-map-config.xml 를 작성한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config 2.0//EN"
    "http://www.ibatis.com/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
<!-- TODO [Step 1-5] iBATIS 의 sql-map-config 설정 파일 작성 -->

<settings useStatementNamespaces="false" cacheModelsEnabled="true" />

<!-- Spring 2.5.5 이상, iBATIS 2.3.2 이상에서는 iBATIS 연동을 위한
SqlMapClientFactoryBean
정의 시 mappingLocations 속성으로 Sql 매핑 파일의 일괄 지정이 가능하다.
("sqlMapClient" bean 설정 시 mappingLocations="classpath:/META-
INF/sqlmap/mappings/lab-*.xml" 로 지정하였음)
단, sql-map-config-2.dtd 에서 sqlMap 요소를 하나 이상 지정하도록 되어 있으므로 아래
의 dummy 매핑 파일을 설정한다. -->
<sqlMap resource="META-INF/sqlmap/mappings/lab-dummy.xml" />
<sqlMap resource="META-INF/sqlmap/mappings/lab-emp.xml" />
<sqlMap resource="META-INF/sqlmap/mappings/lab-emp-cachemodel.xml" />
</sqlMapConfig>
```

❑ 6. ID Generation Service 설정 확인

- /lab204-ibatis/src/test/resources/META-INF/spring/context-idgen.xml 를 확인한다.

```
<!-- [Step 1-6] Id Generation Service 설정 -->
<bean name="primaryTypeSequenceIds"
class="egovframework.rte.fdl.idgnr.impl.EgovSequenceIdGnrService"
destroy-method="destroy">
<property name="dataSource" ref="dataSource" />
<property name="query" value="SELECT NEXT VALUE FOR empseq FROM DUAL" />
</bean>
```

- 여기서는 Hsqldb 를 사용하여 Oracle 의 DUAL 테이블 역할을 할 수 있도록 초기화 스크립트 sql 에 create 하였으며, DB Sequence 기반의 Id Generation 을 사용한 예이다. (위에서 select next value for seq_id from xx 는 Hsqldb 의 특화된 sequence 사용 문법임에 유의!)

❑ 7. common설정 확인

- /lab204-ibatis/src/test/resources/META-INF/spring/context-common.xml 을 **확인**한다.

- **PropertyPlaceholderConfigurer 설정 :** 외부 properties 파일을 Container 구동 시 미리 Spring Bean 설정 파일의 속성값으로 대체하여 처리해주는 PropertyPlaceholderConfigurer 설정
- **MessageSource 설정 :** Locale 에 따른 다국어 처리를 쉽게 해주는 messageSource 설정. 여기서는 전자정부 실행환경의 id generation 서비스와 properties 서비스의 메시지 파일과 업무 어플리케이션을 위한 사용자 메시지(/message/message-common - message-common_en_US.properties, message-common_ko_KR.properties 를 확인할 것) 를 지정하였다.
- **전자정부 TraceHandler 설정 관련 :** exception 처리 Handler 와 유사하게 특정한 상황에서 사용자가 Trace Handler 를 지정하여 사용할 수 있도록 전자정부 프레임워크에서 가이드하고 있는 TraceHandler 설정
- **component-scan 설정 :** 스테레오 타입 Annotation 을 인식하여 Spring bean 으로 자동 등록하기 위한 component-scan 설정

❑ 8. aspect 설정 확인 (1/2)

- /lab204-ibatis/src/test/resources/META-INF/spring/context-aspect.xml 를 확인한다.

```
<aop:config>
	<aop:pointcut id="serviceMethod"
	expression="execution(* egovframework.lab..impl.*Impl.*(..))" />

	<aop:aspect ref="exceptionTransfer">
		<aop:after-throwing throwing="exception"
		pointcut-ref="serviceMethod" method="transfer" />
	</aop:aspect>
</aop:config>

<bean id="exceptionTransfer" class="egovframework.rte.fdl.cmmn.aspect.ExceptionTransfer">
	<property name="exceptionHandlerService">
		<list>
		<ref bean="defaultExceptionHandleManager" />
		</list>
	</property>
</bean>

... (계속)
```

Step 1. 구동 환경 설정

❑ 8. aspect 설정 확인 (2/2)

```
... (이어서)

<bean id="defaultExceptionHandleManager"
class="egovframework.rte.fdl.cmmn.exception.manager.DefaultExceptionHandleManager">
      <property name="reqExpMatcher" ref="antPathMater" />
      <property name="patterns">
            <list>
            <value>**service.impl.*</value>
            </list>
      </property>
      <property name="handlers">
            <list>
            <ref bean="egovHandler" />
            </list>
      </property>
</bean>


<bean id="egovHandler"
class="egovframework.lab.dataaccess.common.JdbcLoggingExcepHndlr" />
```

– Spring AOP(xml 설정 방식) 를 사용하여 비지니스 메서드에서 exception 이 발생한 경우 일괄적으로 ExceptionTransfer 의 transfer 메서드 기능(Advice) 를 수행해 주게 됨. --> Exception logging 및 BizException 형태로 wrapping 하여 재처리하는 Exception 공통처리 후 ExceptionHandleManager 에 의해 관리(설정) 되는 Handler (ex. exception 내용을 메일링 한더던지.. 사용자 구현 가능) 가 자동적으로 추가 수행될 수 있음.

Step 2. 자바 클래스 작성

❑ 1. Service Interface 확인

- /lab204-ibatis/src/main/java/egovframework/lab/dataaccess/service/EmpService.java 를 **확인**한다.

❑ 2. VO 확인

- /lab204-ibatis/src/main/java/egovframework/lab/dataaccess/service/EmpVO.java 를 **확인**한다.
- (참고) Getter와 Setter 메서드는 다음과 같이 생성할 수 있다.

  우클릭 > Source > Generate getters and setters > Select All 선택 > OK

Step 2. 자바 클래스 작성

❑ 3. Annotation을 적용한 Impl 작성

– /lab204-ibatis/src/main/java/egovframework/lab/dataaccess/service/impl/EmpServiceImpl.java 에 아래 내용을 추가한다.

```
@Service("empService")
public class EmpServiceImpl extends EgovAbstractServiceImpl implements EmpService {

    // TODO [Step 2-3] EmpServiceImpl 작성 추가

    @Resource(name = "primaryTypeSequenceIds")
    EgovIdGnrService egovIdGnrService;

    @Resource(name = "empDAO")
    private EmpDAO empDAO;
```

– Ctrl + Shift + O 를 눌러서 import 되지 않은 클래스를 import 시킨다.

– 위 방법을 통해서도 import 되지 않은 클래스가 남아있으면, 이클립스 상단에 Project > Clean을 수행한다.

Step 2. 자바 클래스 작성

❑ 4. DAO 작성

- /lab204-ibatis/src/main/java/egovframework/lab/dataaccess/service/impl/EmpDAO.java 를 작성한다.

```
@Repository("empDAO")
public class EmpDAO extends EgovAbstractDAO {

// TODO [Step 2-4] EmpDAO 작성
    public void insertEmp(EmpVO vo) {
        insert("insertEmp", vo);
    }
    public int updateEmp(EmpVO vo) {
        return update("updateEmp", vo);
    }
    public int deleteEmp(EmpVO vo) {
        return delete("deleteEmp", vo);
    }
    public EmpVO selectEmp(EmpVO vo) {
        return (EmpVO) select ("selectEmp", vo);
        //return (EmpVO) select ("selectEmpUsingCacheModelLRU", vo);
    }
    @SuppressWarnings("unchecked")
    public List<EmpVO> selectEmpList(EmpVO searchVO) {
        return (List<EmpVO>) list("selectEmpList", searchVO);
    }
}
```

Step 2. SQL 매핑 파일 작성

❑ 5. mapping xml 작성 (1/6)

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-emp.xml 를 작성한다.

```
<sqlMap namespace="Emp">
<!-- TODO [Step 2-5] lab-emp.xml mapping xml 작성 -->

<typeAlias alias="empVO" type="egovframework.lab.dataaccess.service.EmpVO" />

<resultMap id="empResult" class="empVO">
<result property="empNo" column="EMP_NO" />
<result property="empName" column="EMP_NAME" />
<result property="job" column="JOB" />
<result property="mgr" column="MGR" />
<result property="hireDate" column="HIRE_DATE" />
<result property="sal" column="SAL" />
<result property="comm" column="COMM" />
<result property="deptNo" column="DEPT_NO" />
</resultMap>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 5. mapping xml 작성 (2/6)

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-emp.xml 를 작성한다.

```
<insert id="insertEmp" parameterClass="empVO">
<![CDATA[
insert into EMP
   (EMP_NO,
    EMP_NAME,
    JOB,
    MGR,
    HIRE_DATE,
    SAL,
    COMM,
    DEPT_NO)
    values   (#empNo#,
    #empName#,
    #job#,
    #mgr#,
    #hireDate#,
    #sal#,
    #comm#,
    #deptNo#)
]]>
</insert>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 5. mapping xml 작성 (3/6)

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-emp.xml 를 작성한다.

```
<update id="updateEmp" parameterClass="empVO">
<![CDATA[
update EMP
   set EMP_NAME= #empName#,
   JOB = #job#,
   MGR = #mgr#,
   HIRE_DATE = #hireDate#,
   SAL = #sal#,
   COMM = #comm#,
   DEPT_NO = #deptNo#
 where EMP_NO = #empNo#
]]>
</update>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 5. mapping xml 작성 (4/6)

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-emp.xml 를 작성한다.

```
<delete id="deleteEmp" parameterClass="empVO">
<![CDATA[
delete from EMP
 where EMP_NO = #empNo#
]]>
</delete>

<select id="selectEmp" parameterClass="empVO" resultMap="empResult">
<![CDATA[
select EMP_NO,
   EMP_NAME,
   JOB,
   MGR,
   HIRE_DATE,
   SAL,
   COMM,
   DEPT_NO
  from EMP
 where EMP_NO = #empNo#
]]>
</select>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 5. mapping xml 작성 (5/6)

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-emp.xml 를 작성한다.

```
<select id="selectEmpList" parameterClass="empVO" resultMap="empResult">
<![CDATA[
select EMP_NO,
   EMP_NAME,
   JOB,
   MGR,
   HIRE_DATE,
   SAL,
   COMM,
   DEPT_NO
  from EMP
 where 1 = 1
]]>
<isNotNull prepend="and" property="empNo">
EMP_NO = #empNo#
</isNotNull>
<isNotNull prepend="and" property="empName">
EMP_NAME LIKE '%' || #empName# || '%'
</isNotNull>
</select>

</sqlMap>
```

Step 2. SQL 매핑 파일 작성

❑ 5. mapping xml 작성 (6/6)

- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-dummy.xml 를 확인한다.
- /lab204-ibatis/src/test/resources/META-INF/sqlmap/mappings/lab-emp-cachemodel.xml 를 확인한다.

Step 3. 테스트 케이스 확인

❑ 1. EmpServiceTest 확인

– /lab204-ibatis/src/test/java/egovframework/lab/dataaccess/service/EmpServiceTest.java 를 확인한다.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = {"classpath*:META-INF/spring/context-*" })
@TransactionConfiguration(transactionManager = "txManager", defaultRollback = false)
@Transactional
public class EmpServiceTest {

    // TODO [Step 3-1] BasicCRUDTest 실행

    @Resource(name = "dataSource")
    DataSource dataSource;

    @Resource(name = "empService")
    EmpService empService;

    @Before
    public void onSetUp() throws Exception {
        // 테스트 편의상 매 테스트메서드 수행전 외부의 sql file 로부터 DB 초기화
        // (기존 테이블 삭제/생성)
        JdbcTestUtils.executeSqlScript(new JdbcTemplate(dataSource), new ClassPathResource("META-
INF/testdata/sample_schema_hsql.sql"), true);
    }


    public EmpVO makeVO() throws ParseException {
        EmpVO vo = new EmpVO();

        // empNo 는 Biz. 서비스 내에서 id generation service 에
        // 의해 key 를 따고 설정할 것임.

        ... (계속)
```

Step 3. 테스트 케이스 확인

```
        ... (이어서)
        vo.setEmpName("홍길동");
        vo.setJob("개발자");
        vo.setMgr(new BigDecimal(7902));
        SimpleDateFormat sdf =
            new SimpleDateFormat("yyyy-MM-dd", java.util.Locale.getDefault());
        vo.setHireDate(sdf.parse("2009-07-09"));
        vo.setSal(new BigDecimal(1000));
        vo.setComm(new BigDecimal(0));
        vo.setDeptNo(new BigDecimal(20));

        return vo;
    }

    public void checkResult(EmpVO vo, EmpVO resultVO) {
        assertNotNull(resultVO);
        assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
        assertEquals(vo.getEmpName(), resultVO.getEmpName());
        assertEquals(vo.getJob(), resultVO.getJob());
        assertEquals(vo.getMgr(), resultVO.getMgr());
        assertEquals(vo.getHireDate(), resultVO.getHireDate());
        assertEquals(vo.getSal(), resultVO.getSal());
        assertEquals(vo.getComm(), resultVO.getComm());
        assertEquals(vo.getDeptNo(), resultVO.getDeptNo());
    }

    @Test
    public void testInsertEmp() throws Exception {
        EmpVO vo = makeVO();
        // insert
        BigDecimal empNo = empService.insertEmp(vo);
        vo.setEmpNo(empNo);
        // select
        EmpVO resultVO = empService.selectEmp(vo);
        // check
        checkResult(vo, resultVO);
    }
     ... (계속)
```

Step 3. 테스트 케이스 확인

```
    ... (이어서)
@Test
public void testUpdateEmp() throws Exception {
    EmpVO vo = makeVO();

    // insert
    BigDecimal empNo = empService.insertEmp(vo);
    vo.setEmpNo(empNo);

    // data change
    vo.setEmpName("홍길순");
    vo.setJob("설계자");

    // update
    empService.updateEmp(vo);

    // select
    EmpVO resultVO = empService.selectEmp(vo);

    // check
    checkResult(vo, resultVO);
}

@Test
public void testDeleteEmp() throws Exception {
    EmpVO vo = makeVO();

    // insert
    BigDecimal empNo = empService.insertEmp(vo);
    vo.setEmpNo(empNo);

    // delete
    empService.deleteEmp(vo);

    ... (계속)
```

Step 3. 테스트 케이스 확인

```
        ... (이어서)
        // select
        try {
            @SuppressWarnings("unused")
            EmpVO resultVO = empService.selectEmp(vo);
            fail("EgovBizException 이 발생해야 합니다.");
        } catch (Exception e) {
            assertNotNull(e);
            // 여기서는 비지니스 단에서 명시적으로 exception 처리 하였음.
            // AbstractServiceImpl 을 extends 하고
            // processException("info.nodata.msg"); 과 같이 메서드 콜 형태로 처리
            assertTrue(e instanceof EgovBizException);
            assertEquals("info.nodata.msg", ((EgovBizException) e)
                .getMessageKey());
            assertEquals("해당 데이터가 없습니다.", e.getMessage());
        }
    }

    @Test
    public void testSelectEmpList() throws Exception {
        EmpVO vo = makeVO();
        // insert
        BigDecimal empNo = empService.insertEmp(vo);
        vo.setEmpNo(empNo);

        // 검색조건으로 key 설정
        EmpVO searchVO = new EmpVO();
        searchVO.setEmpNo(vo.getEmpNo());

        // selectList
        List<EmpVO> resultList = empService.selectEmpList(searchVO);

        ... (계속)
```

Step 3. 테스트 케이스 확인

```
        ... (이어서)
        // key 조건에 대한 결과는 한건일 것임
        assertNotNull(resultList);
        assertTrue(resultList.size() > 0);
        assertEquals(1, resultList.size());
        checkResult(vo, resultList.get(0));

        // 검색조건으로 name 설정 - '%' || #empName# || '%'
        EmpVO searchVO2 = new EmpVO();
        searchVO2.setEmpName(""); // '%' || '' || '%'
                                  // --> '%%'
        // selectList
        List<EmpVO> resultList2 = empService.selectEmpList(searchVO2);

        // like 조건에 대한 결과는 한건 이상일 것임
        assertNotNull(resultList2);
        assertTrue(resultList2.size() > 0);
    }
}
```

Step 3. 테스트 케이스 확인

❑ 2. CacheModelTest 확인

- /lab204-ibatis/src/test/java/egovframework/lab/dataaccess/service/CacheModelTest.java 를 확인한다.
- iBatis 캐시 기능을 테스트 하기 위해서는 아래와 같이 **DAO 클래스의 주석을 변경한 후에** 진행한다.
- Statement id가 selectEmpUsingCacheModelLRU인 select문에는 캐시 기능이 설정되어 있다.
  - `<select id="selectEmpUsingCacheModelLRU" … cacheModel="cacheEmpLRU">`

```
// EmpDAO 클래스 주석 변경할 부분
public EmpVO selectEmp(EmpVO vo) {
// return (EmpVO) select ("selectEmp", vo);
return (EmpVO) select("selectEmpUsingCacheModelLRU", vo);
}
```

[참고] Logging 설정 확인

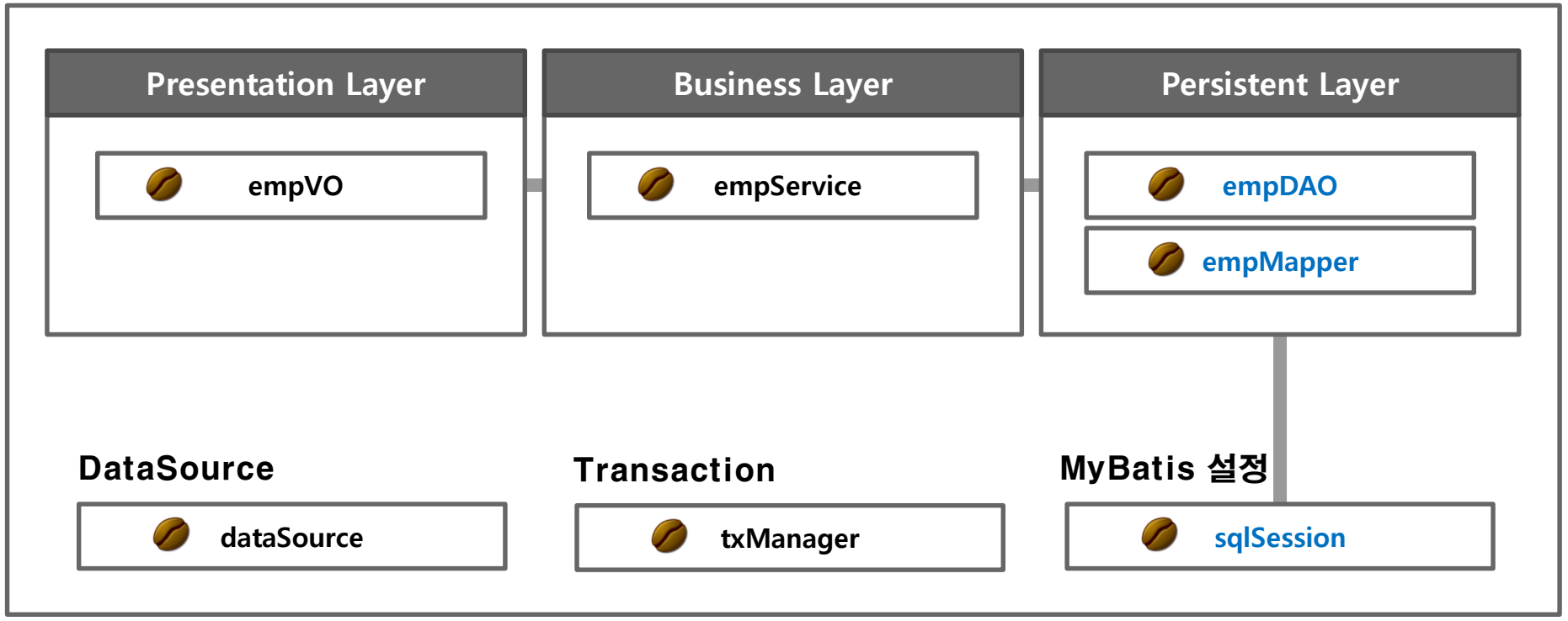
❑ Logger 및 로그레벨 설정

– /lab204-ibatis/src/test/resources/log4j2.xml 에 다음 Logger 를 확인한다.

```
<!-- log SQL with timing information, post execution -->
<Logger name="jdbc.sqltiming" level="INFO" additivity="false">
<AppenderRef ref="console2" />
</Logger>
<Logger name="jdbc.sqlonly" level="INFO" additivity="false">
<AppenderRef ref="console2" />
</Logger>
```

LAB 205-MyBatis 실습

❑ lab205-mybatis 프로젝트의 beans



Step 1. 구동 환경 설정

❑ 1. Hsqldb 초기화 스크립트

- /lab205-mybatis/src/test/resources/META-INF/testdata/sample_schema_hsql.sql 를 확인한다.
- 현 실습 프로젝트에서는 편의상 매 테스트 케이스 재실행 시 관련 table 을 drop/create 하고 있음.

❑ 2. dataSource 설정 (테스트 편의성을 위해 memory DB 사용)

- <bean id="dataSource" .../>를 이용한 방법
- <jdbc:embedded-database .../>를 이용한 방법

❑ 2. dataSource 설정

- /lab205-mybatis/src/test/resources/META-INF/spring/context-datasource.xml 에 dataSource 빈 설정을 추가한다.
- 스프링에서 제공하는 EmbeddedDataBase를 생성하여 테스트한다.

```
<!--  TODO [Step 1-2] DataSource 설정 -->
<jdbc:embedded-database id="dataSource" type="HSQL">
<jdbc:script location= "META-INF/testdata/sample_schema_hsql.sql"/>
</jdbc:embedded-database>
```

❑ 3. transaction 설정

- /lab205-mybatis/src/test/resources/META-INF/spring/context-transaction.xml 를 작성한다.

```
<!-- TODO [Step 1-3] transaction 설정 -->
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>
<tx:annotation-driven transaction-manager="txManager" />
```

cf.) 여기서는 transaction manager 와 메소드 혹은 클래스 레벨에 @Transactional 을 선언하여 트랜잭션 서비스를 이용한다. @Transactional Anntation 스캔을 위해서는 <tx:annotation-driven />을 선언해야 한다. @Transactional을 이용하면 대상 메소드에 개별적으로 트랜잭션을 지정할 수 있다는 장점이 있으나, 보통 AOP 형식(tx:aop)으로 선언하여 트랜잭션 대상 메서드들에 일괄 지정하는 경우가 많다.

❑ 4. Spring 의 MyBatis 연동 설정

- /lab205-mybatis/src/test/resources/META-INF/spring/context-mybatis.xml 를 작성한다.

```
<!-- TODO [Step 1-4] MyBaits와 Spring 연동 설정 -->
<bean id="sqlSession" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
<property name="configLocation" value="classpath:/META-INF/sqlmap/sql-mybatis-config.xml" />
<!-- <property name="mapperLocations" value="classpath:**/lab-*.xml" /> -->
</bean>
```

- 최신 프레임워크 환경에서는 sql-mybatis-config.xml 내에 개별 sql 맵핑 파일을 일일이 지정하는 것이 아니라, 위의 mapperLocations 영역을 주석 해제하여 Spring 의 ResourceLoader 형식으로 패턴 매칭에 의거한 일괄 로딩으로 처리가 가능하다. (단, 테스트 결과 CacheModel 등의 일부 기능에서 문제가 발생하는 경우가 있으므로 사용에 유의할것!!)

❑ 5. MyBatis 의 sql-mybatis-config 설정 파일 작성

- /lab205-mybatis/src/test/resources/META-INF/sqlmap/sql-mybatis-config.xml 를 작성한다.

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

<configuration>
<!-- TODO [Step 1-5] MyBatis Configuration File 작성  -->
<typeAliases>
<typeAlias alias="empVO" type="egovframework.lab.dataaccess.service.EmpVO" />
</typeAliases>

<!-- MyBatis 연동을 위한 SqlSessionFactoryBean 정의 시 mapperLocations 속성으로
한 번에 모든 Mapper XML File을 설정할 수 있다.
(<property name="mapperLocations" value="classpath:**/lab-*.xml" /> 추가)
단, 아래 <mappers> 설정과 mapperLocations 설정 중 한가지만 선택해야 한다.
-->
<mappers>
<mapper resource="META-INF/sqlmap/mappers/lab-dao-class.xml" />
<mapper resource="META-INF/sqlmap/mappers/lab-mapper-interface.xml" />
</mappers>

</configuration>
```

❑ 6. ID Generation Service 설정 확인

- /lab205-mybatis/src/test/resources/META-INF/spring/context-idgen.xml 를 **확인**한다.

```
<!-- [Step 1-6] Id Generation Service 설정 -->
<bean name="primaryTypeSequenceIds"
class="egovframework.rte.fdl.idgnr.impl.EgovSequenceIdGnrService"
destroy-method="destroy">
<property name="dataSource" ref="dataSource" />
<property name="query" value="SELECT NEXT VALUE FOR empseq FROM DUAL" />
</bean>
```

- 여기서는 Hsqldb 를 사용하여 Oracle 의 DUAL 테이블 역할을 할 수 있도록 초기화 스크립트 sql 에 create 하였으며, DB Sequence 기반의 Id Generation 을 사용한 예이다. (위에서 select next value for seq_id from xx 는 Hsqldb 의 특화된 sequence 사용 문법임에 유의!)

❑ 7. common설정 확인

- /lab205-mybatis/src/test/resources/META-INF/spring/context-common.xml 을 **확인**한다.

- **PropertyPlaceholderConfigurer 설정 :** 외부 properties 파일을 Container 구동 시 미리 Spring Bean 설정 파일의 속성값으로 대체하여 처리해주는 PropertyPlaceholderConfigurer 설정
- **MessageSource 설정 :** Locale 에 따른 다국어 처리를 쉽게 해주는 messageSource 설정. 여기서는 전자정부 실행환경의 id generation 서비스와 properties 서비스의 메시지 파일과 업무 어플리케이션을 위한 사용자 메시지(/message/message-common - message-common_en_US.properties, message-common_ko_KR.properties 를 확인할 것) 를 지정하였다.
- **전자정부 TraceHandler 설정 관련 :** exception 처리 Handler 와 유사하게 특정한 상황에서 사용자가 Trace Handler 를 지정하여 사용할 수 있도록 전자정부 프레임워크에서 가이드하고 있는 TraceHandler 설정
- **component-scan 설정 :** 스테레오 타입 Annotation 을 인식하여 Spring bean 으로 자동 등록하기 위한 component-scan 설정

❑ 8. aspect 설정 확인 (1/2)

- /lab205-mybatis/src/test/resources/META-INF/spring/context-aspect.xml 를 확인한다.

```
<aop:config>
	<aop:pointcut id="serviceMethod"
	expression="execution(* egovframework.lab..impl.*Impl.*(..))" />

	<aop:aspect ref="exceptionTransfer">
		<aop:after-throwing throwing="exception"
		pointcut-ref="serviceMethod" method="transfer" />
	</aop:aspect>
</aop:config>

<bean id="exceptionTransfer" class="egovframework.rte.fdl.cmmn.aspect.ExceptionTransfer">
	<property name="exceptionHandlerService">
		<list>
		<ref bean="defaultExceptionHandleManager" />
		</list>
	</property>
</bean>

... (계속)
```

Step 1. 구동 환경 설정

❑ 8. aspect 설정 확인 (2/2)

```
... (이어서)

<bean id="defaultExceptionHandleManager"
class="egovframework.rte.fdl.cmmn.exception.manager.DefaultExceptionHandleManager">
      <property name="reqExpMatcher" ref="antPathMater" />
      <property name="patterns">
            <list>
            <value>**service.impl.*</value>
            </list>
      </property>
      <property name="handlers">
            <list>
            <ref bean="egovHandler" />
            </list>
      </property>
</bean>


<bean id="egovHandler"
class="egovframework.lab.dataaccess.common.JdbcLoggingExcepHndlr" />
```

– Spring AOP(xml 설정 방식) 를 사용하여 비지니스 메서드에서 exception 이 발생한 경우 일괄적으로 ExceptionTransfer 의 transfer 메서드 기능(Advice) 를 수행해 주게 됨. --> Exception logging 및 BizException 형태로 wrapping 하여 재처리하는 Exception 공통처리 후 ExceptionHandleManager 에 의해 관리(설정) 되는 Handler (ex. exception 내용을 메일링 한더던지.. 사용자 구현 가능) 가 자동적으로 추가 수행될 수 있음.

Step 2. 자바 클래스 작성

❑ 1. Annotation을 적용한 Impl 작성

– /lab205-mybatis/src/main/java/egovframework/lab/dataaccess/service/impl/EmpServiceImpl.java 에 아래 내용을 추가한다.

```
@Service("empService")
public class EmpServiceImpl extends EgovAbstractServiceImpl implements EmpService {

    // TODO [Step 2-1] EmpServiceImpl 작성 추가

    @Resource(name = "empDAO")
    private EmpDAO empDAO;
    // TODO [Step 4-2] EmpServiceImpl의 주석 EmpMapper -> EmpDAO 변경 후 다시 테스트

    // TODO [Step 3-1] EmpServiceImpl 변경
    // EmpMapper를 사용하도록 주석 변경
    // @Resource(name = "empMapper")
    // EmpMapper empDAO;
```

– Ctrl + Shift + O 를 눌러서 import 되지 않은 클래스를 import 시킨다.

– 위 방법을 통해서도 import 되지 않은 클래스가 남아있으면, 이클립스 상단에 Project > Clean을 수행한다.

Step 2. 자바 클래스 작성

❑ 2. DAO 작성

- /lab205-mybatis/src/main/java/egovframework/lab/dataaccess/service/impl/EmpDAO.java 를 작성한다.

```
@Repository("empDAO")
public class EmpDAO extends EgovAbstractMapper {
// TODO [Step 2-2] EmpDAO 작성 (EgovAbstractMapper 상속한 DAO)

    public void insertEmp(EmpVO vo) {
        insert("Emp.insertEmp", vo);
    }
    public int updateEmp(EmpVO vo) {
        return update("Emp.updateEmp", vo);
    }
    public int deleteEmp(EmpVO vo) {
        return delete("Emp.deleteEmp", vo);
    }
    public EmpVO selectEmp(EmpVO vo) {
        return selectOne("Emp.selectEmp", vo);
    }
    @SuppressWarnings("unchecked")
    public List<EmpVO> selectEmpList(EmpVO searchVO) {
        return selectList("Emp.selectEmpList", searchVO);
    }
}
```

Step 2. SQL 매핑 파일 작성

❑ 3. mapping xml 작성 (1/4)

- /lab205-mybatis/src/test/resources/META-INF/sqlmap/mappers/lab-dao-class.xml 를 작성한다.

```
<?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="Emp">

<!-- TODO [Step 2-3] lab-dao-class.xml 작성 (EgovAbstractMapper 상속한 DAO) -->
<resultMap id="empResult" type="empVO">
<id property="empNo" column="EMP_NO" />
<result property="empName" column="EMP_NAME" />
<result property="job" column="JOB" />
<result property="mgr" column="MGR" />
<result property="hireDate" column="HIRE_DATE" />
<result property="sal" column="SAL" />
<result property="comm" column="COMM" />
<result property="deptNo" column="DEPT_NO" />
</resultMap>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 3. mapping xml 작성 (2/4)

- /lab205-mybatis/src/test/resources/META-INF/sqlmap/mappers/lab-dao-class.xml 를 작성한다.

```
<insert id="insertEmp" parameterType="empVO">
<![CDATA[
insert into EMP (EMP_NO, EMP_NAME, JOB, MGR, HIRE_DATE, SAL, COMM, DEPT_NO)
values(#{empNo}, #{empName}, #{job}, #{mgr}, #{hireDate}, #{sal}, #{comm}, #{deptNo})
]]>
</insert>

<update id="updateEmp" parameterType="empVO">
<![CDATA[
update EMP
set EMP_NAME = #{empName},
JOB = #{job},
MGR = #{mgr},
HIRE_DATE = #{hireDate},
SAL = #{sal},
COMM = #{comm},
DEPT_NO = #{deptNo}
where EMP_NO = #{empNo}
]]>
</update>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 3. mapping xml 작성 (3/4)

- /lab205-mybatis/src/test/resources/META-INF/sqlmap/mappers/lab-dao-class.xml 를 작성한다.

```
<delete id="deleteEmp" parameterType="empVO">
<![CDATA[
delete from EMP
where EMP_NO = #{empNo}
]]>
</delete>

<select id="selectEmp" parameterType="empVO" resultMap="empResult">
<![CDATA[
select EMP_NO, EMP_NAME, JOB, MGR, HIRE_DATE, SAL, COMM, DEPT_NO
from EMP
where EMP_NO = #{empNo}
]]>
</select>
```

- 다음 슬라이드에서 계속...

Step 2. SQL 매핑 파일 작성

❑ 3. mapping xml 작성 (4/4)

- /lab205-mybatis/src/test/resources/META-INF/sqlmap/mappers/lab-dao-class.xml 를 작성한다.

```
<select id="selectEmpList" parameterType="empVO" resultMap="empResult">
<![CDATA[
Select EMP_NO, EMP_NAME, JOB, MGR, HIRE_DATE, SAL, COMM, DEPT_NO
From EMP
where 1 = 1
]]>
<if test="empNo != null">
AND EMP_NO = #{empNo}
</if>
<if test="empName != null">
AND EMP_NAME LIKE '%' || #{empName} || '%'
</if>
</select>
</mapper>
```

Step 3. 자바 클래스 작성

❑ 1. Service Impl 추가 변경 작성

- /lab205-mybatis/src/main/java/egovframework/lab/dataaccess/service/impl/EmpServiceImpl.java 를 작성한다.
- Mapper Interface 방식으로 테스트 시 사용할 DAO 클래스를 미리 작성한다.

  (주석처리를 하여 EmpDAO empDAO와 충돌되지 않도록 한다.)

```
@Service("empService")
public class EmpServiceImpl extends EgovAbstractServiceImpl implements EmpService {

    ...

    // TODO [Step 3-1] EmpServiceImpl 추가 작성
    // EmpMapper를 사용하도록 주석 변경
    // @Resource(name = "empMapper")
    // EmpMapper empDAO;
```

Step 3. 자바 클래스 작성

❑ 2. Mapper Interface 작성

- /lab205-mybatis/src/main/java/egovframework/lab/dataaccess/service/impl/EmpMapper.java 를 작성한다.
- 아래와 같이 MyBatis에서는 Dao 클래스 대신 Interface를 이용해 데이터 처리가 가능하다.

```
@Mapper("empMapper")
public interface EmpMapper {

    // TODO [Step 3-2] EmpMapper 작성 (Mapper Interface)

    public void insertEmp(EmpVO vo);

    public int updateEmp(EmpVO vo);

    public int deleteEmp(EmpVO vo);

    public EmpVO selectEmp(EmpVO vo);

    public List<EmpVO> selectEmpList(EmpVO searchVO);

}
```

Step 3. 자바 클래스 작성

❑ 3. @Mapper 스캔 설정

- /lab205-mybatis/src/test/resources/META-INF/spring/context-mybatis.xml 에 다음을 추가한다.

```
<!-- MapperConfigurer setup for @Mapper -->
<!-- TODO [Step 3-3] MyBatis의 Mapper Interface 자동스캔 설정 -->

<bean class="egovframework.rte.psl.dataaccess.mapper.MapperConfigurer">
<property name="basePackage" value="egovframework.lab.dataaccess.service.impl" />
</bean>
```

Step 3. SQL 매핑 파일 작성

❑ 4. mapping xml 작성

- /lab205-mybatis/src/test/resources/META-INF/sqlmap/mappers/lab-mapper-interface.xml 를 작성한다.
- /lab-dao-class.xml과 <mapper>의 name 속성값만 다르다.

```
<mapper namespace="egovframework.lab.dataaccess.service.impl.EmpMapper">

<!-- TODO [Step 3-4] lab-mapper-interface.xml 작성 (Mapper Interface)
실습교재 p.45(<resultMap>부터) ~ p.48까지 내용을 동일하게 작성
-->
```

- DAO 클래스의 Statement 호출 방식: 사용자가 직접 지정해준 ID 파라미터 값과 일치하는 Statement를 호출. 동일한 Statement ID가 있으면, <mapper>의 namespace를 지정한다.
  - namespace=A, statement id=insertEmp → A.insertEmp으로 호출
  - namespace=B, statement id=insertEmp → B.insertEmp으로 호출
- Mapper 인터페이스의 Statement 호출 방식: 메소드명과 일치하는 Statement를 자동 호출. 이 때 MyBatis는 호출된 메서드가 포함된 인터페이스의 풀네임을 namespace 값으로 사용하기 때문에, 반드시 namesapce 값을 지정해주어야 한다.
  - namespace=x.y.z.EmpMapper, statement id=insertEmp → 내부적으로 x.y.z.EmpMapper.insertEmp을 호출

Step 4. 테스트 케이스 확인

❑ 1. EmpServiceTest 확인 – EmpDAO 테스트

– /lab205-mybatis/src/test/java/egovframework/lab/dataaccess/service/EmpServiceTest.java 를 확인하고 실행한다.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath*:META-INF/spring/context-*" })
@TransactionConfiguration(transactionManager = "txManager", defaultRollback = false)
@Transactional
public class EmpServiceTest {

// TODO [Step 4-1] EmpServiceTest 실행
@Resource(name = "dataSource")
DataSource dataSource;

@Resource(name = "empService")
EmpService empService;

@Before
public void onSetUp() throws Exception {
// 편의상 각 테스트 메서드 수행 전에
// 외부의 스크립트 파일(sample_schema_hsql.sql)로 DB를 초기화하도록 설정
JdbcTestUtils.executeSqlScript(new JdbcTemplate(dataSource), new ClassPathResource("META-
INF/testdata/sample_schema_hsql.sql"), true);
}
```

Step 4. 테스트 케이스 확인

```
/**
 * 사원정보 생성
 *
 * @throws ParseException
 */
public EmpVO makeVO() throws ParseException {
EmpVO vo = new EmpVO();

// empNo는 Biz. 서비스 내에서 IDGeneration Service 에 의해 key를 생성하고 설정
vo.setEmpName("홍길동");
vo.setJob("개발자");
vo.setMgr(new BigDecimal(7902));
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd",
java.util.Locale.getDefault());
vo.setHireDate(sdf.parse("2009-07-09"));
vo.setSal(new BigDecimal(1000));
vo.setComm(new BigDecimal(0));
vo.setDeptNo(new BigDecimal(20));

return vo;
}

public void checkResult(EmpVO vo, EmpVO resultVO) {
assertNotNull(resultVO);
assertEquals(vo.getEmpNo(), resultVO.getEmpNo());
assertEquals(vo.getEmpName(), resultVO.getEmpName());
assertEquals(vo.getJob(), resultVO.getJob());
assertEquals(vo.getMgr(), resultVO.getMgr());
assertEquals(vo.getHireDate(), resultVO.getHireDate());
assertEquals(vo.getSal(), resultVO.getSal());
assertEquals(vo.getComm(), resultVO.getComm());
assertEquals(vo.getDeptNo(), resultVO.getDeptNo());
}
```

Step 4. 테스트 케이스 확인

```
/** 사원정보 입력 */
@Test
public void testInsertEmp() throws Exception {
EmpVO vo = makeVO();

// insert
BigDecimal empNo = empService.insertEmp(vo);
vo.setEmpNo(empNo);

// select
EmpVO resultVO = empService.selectEmp(vo);

// check
checkResult(vo, resultVO);
}
/** 사원정보 수정 */
@Test
public void testUpdateEmp() throws Exception {
EmpVO vo = makeVO();

// insert
BigDecimal empNo = empService.insertEmp(vo);
vo.setEmpNo(empNo);

// data change
vo.setEmpName("홍길순");
vo.setJob("설계자");

// update
empService.updateEmp(vo);

// select
EmpVO resultVO = empService.selectEmp(vo);

// check
checkResult(vo, resultVO);
}
```

Step 4. 테스트 케이스 확인

```
/** 사원정보 삭제 */
@Test
public void testDeleteEmp() throws Exception {
EmpVO vo = makeVO();

// insert
BigDecimal empNo = empService.insertEmp(vo);
vo.setEmpNo(empNo);

// delete
empService.deleteEmp(vo);

// select
try {
@SuppressWarnings("unused")
EmpVO resultVO = empService.selectEmp(vo);
fail("EgovBizException 이 발생해야 합니다.");
} catch (Exception e) {
assertNotNull(e);
// 여기서는 비지니스 단에서 명시적으로 exception 처리하였음
// AbstractServiceImpl 을 extends 하고
// processException("info.nodata.msg"); 과 같이 메서드 콜 형태로 처리
assertTrue(e instanceof EgovBizException);
assertEquals("info.nodata.msg",
((EgovBizException) e).getMessageKey());
assertEquals("해당 데이터가 없습니다.", e.getMessage());
}
}
```

Step 4. 테스트 케이스 확인

```
/** 사원정보 목록조회 */
@Test
public void testSelectEmpList() throws Exception {
EmpVO vo = makeVO();

// insert
BigDecimal empNo = empService.insertEmp(vo);
vo.setEmpNo(empNo);

// 검색조건으로 empNo 설정
EmpVO searchVO = new EmpVO();
searchVO.setEmpNo(vo.getEmpNo());

// selectList
List<EmpVO> resultList = empService.selectEmpList(searchVO);

// empNo 조건에 대한 결과는 1건일 것임
assertNotNull(resultList);
assertTrue(resultList.size() > 0);
assertEquals(1, resultList.size());
checkResult(vo, resultList.get(0));

// 검색조건으로 empName 설정 - '%' || #{empName} || '%'
EmpVO searchVO2 = new EmpVO();
searchVO2.setEmpName(""); // '%' || '' || '%' // --> '%%'

// selectList
List<EmpVO> resultList2 = empService.selectEmpList(searchVO2);

// like 조건에 대한 결과는 1건 이상일 것임
assertNotNull(resultList2);
assertTrue(resultList2.size() > 0);
}

}
```

Step 4. 테스트 케이스 확인

❑ 2. EmpServiceTest 확인 – EmpMapper 테스트

– /lab205-mybatis/src/test/java/egovframework/lab/dataaccess/service/impl/EmpServiceImpl.java 를 다음과 같이 수정한 후, EmpServiceTest를 다시 실행해본다.

```
@Service("empService")
public class EmpServiceImpl extends EgovAbstractServiceImpl implements EmpService {

   // TODO [Step 2-1] EmpServiceImpl 추가 작성
   // @Resource(name = "empDAO")
   // public EmpDAO empDAO;


     // TODO [Step 4-2] EmpServiceImpl 변경
    // EmpMapper를 사용하도록 주석 변경
     @Resource(name = "empMapper")
     EmpMapper empDAO;
```