

[14] Mybatis - I

처음 JSP&Servlet을 이용할 때는 많은 코드양으로 DB에 접근하다가, Spring에서는 JDBC Template을 이용해서 코드양을 많이 줄일 수 있었다. Mybatis를 이용하면 코드를 더 줄일 수 있고, 자바 코드가 아닌 xml 파일로 DB에 액세스한다. MyBatis는 DB접근을 자바가 아닌 xml로 대체하도록 하는 라이브러리이다. SQL 코드를 자바코드와 분리시키는데 목적이 있다.

진행순서

1. pom.xml:필요라이브러리 의존추가
2. web.xml :한글처리. *.do
3. 실행 첫화면 : css 추가해서 테스트
4. db.properties : DB환경설정정보
5. root-context.xml에 dataSource 빈생성
6. DTO
7. xml 생성 :mapper(empList,deptList) + config
8. root-context.xml에 SqlSessionFactoryBean, sessionTemplate(SqlSessionTemplate) 빈생성
9. DAO
10. Service
11. Controller
12. emp.jsp

Dao -> Jdbc template

-> MyBatis (DAO / interface)

-> JPA

1. pom.xml

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.2.8.RELEASE</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.5</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.5</version>
</dependency>
lombok
<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
```

```
<dependency>

  <groupId>org.projectlombok</groupId>

  <artifactId>lombok</artifactId>

  <version>1.18.24</version>

  <scope>provided</scope>

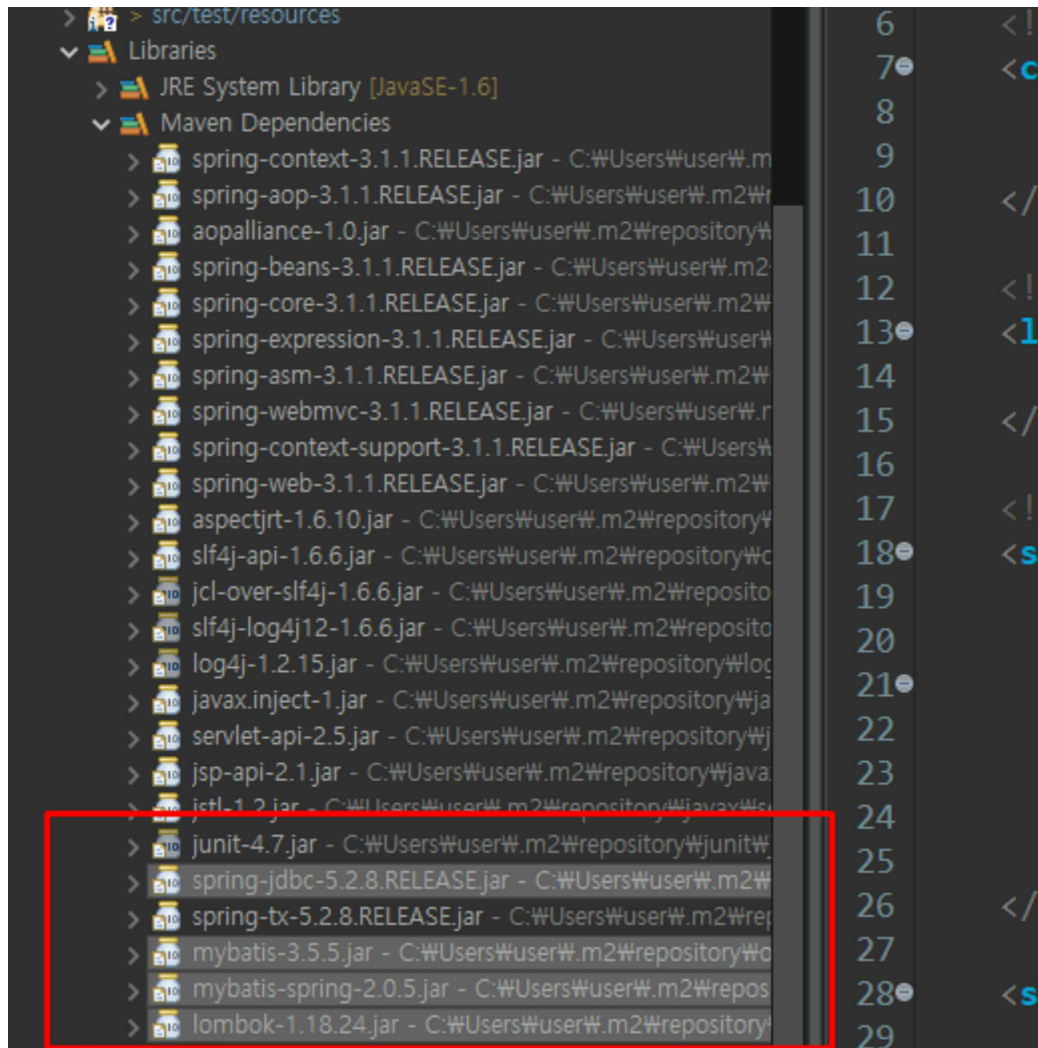
</dependency>
```

2. web.xml

```
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-
class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
```

```
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

- Maven 확인



wep.xml

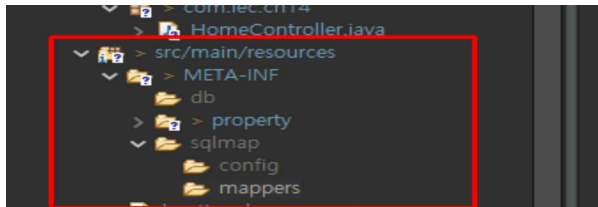
(한글처리) , *.do

```
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
<!-- 한글 필터 추가 -->
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
</web-app>
```

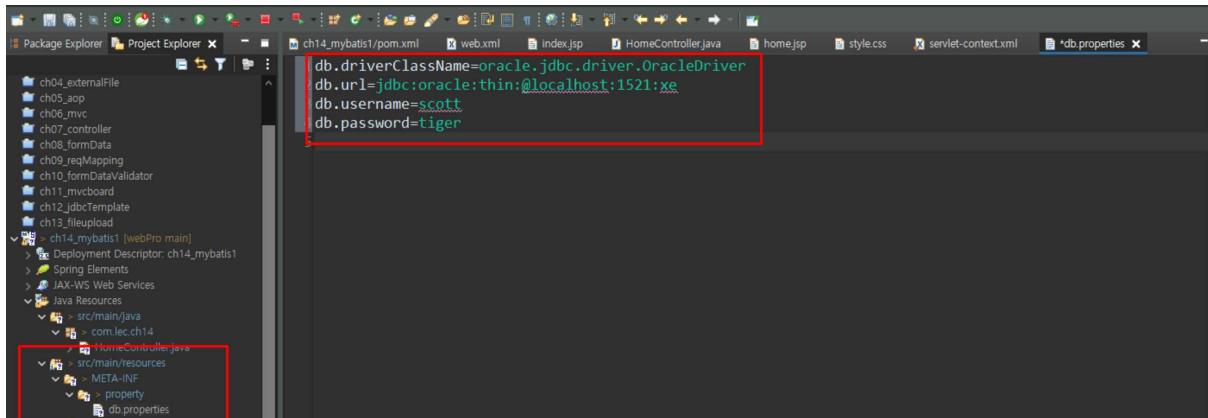
Index.jsp 생성

3. Css

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans:beans xmlns="http://www.springframework.org/schema/mvc"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xmlns:beans="http://www.springframework.org/schema/beans"
5   xmlns:context="http://www.springframework.org/schema/context"
6   xsi:schemaLocation="http://www.springframework.org/schema/mvc https://www.springframework.org/
7     http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/s
8     http://www.springframework.org/schema/context https://www.springframework.org/schema/cont
9
10 <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
11
12 <!-- Enables the Spring MVC @Controller programming model -->
13 <annotation-driven />
14
15 <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in
16 <resources mapping="/resources/**" location="/resources/" />
17 <resources mapping="/css/**" location="/css/" />
18
```



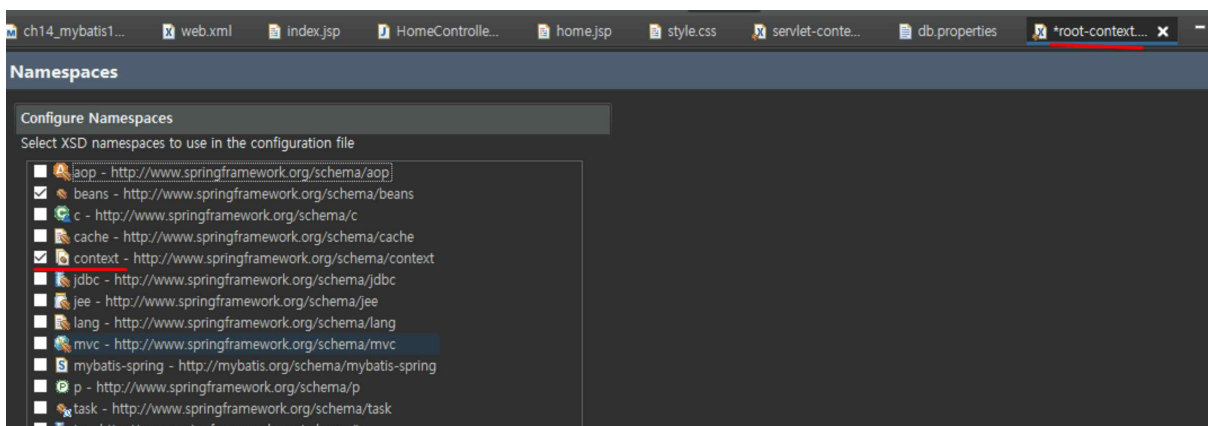
4. db 환경설정정보



src/main/resources/META-INF/property/db.properties

```
db.driverClassName=oracle.jdbc.driver.OracleDriver
db.url=jdbc:oracle:thin:@localhost:1521:xe
db.username=scott
db.password=tiger
```

5. root-context.xml에 dataSource 빈생성

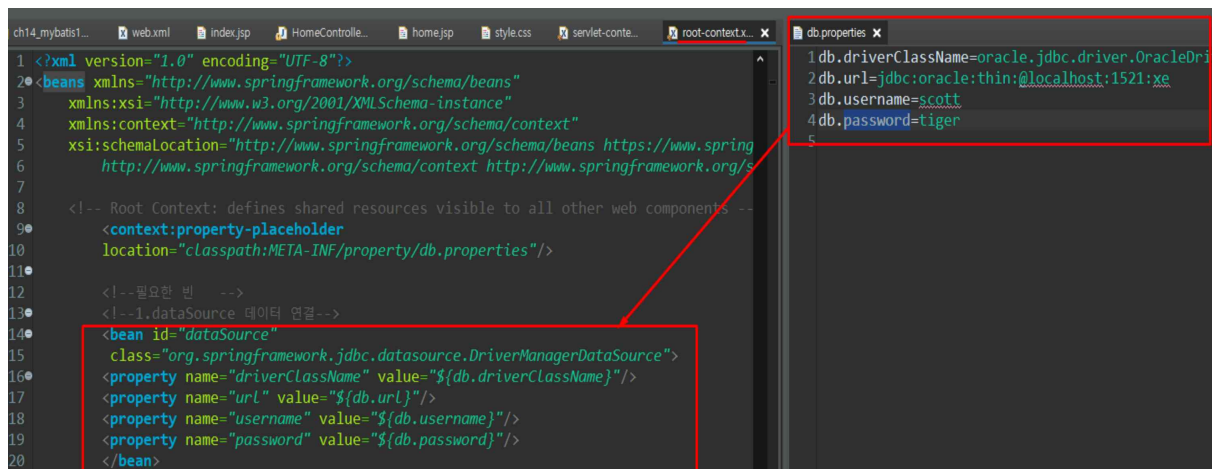


필요한 bean

```
<!--필요한 빈 -->  
<!--1.dataSource 데이터 연결-->  
  
<!-- sqlSessionSessionFactoryBean sql 저장 -->  
  
<!-- FactoryBean 의존 (a 불러와 b 불러와 )  
sqlSessionTemplate -->
```

-DataSource

Class = DriverManagerDataSource 주소



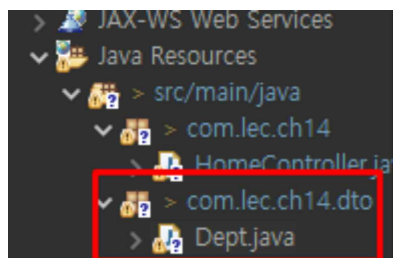
SQL 작성

```
-- Dept.xml 파일안에서 sql문 하나당 id 하나씩 (나중에 sql문을 사용하려 할때 그 sql문의 아이디를 호출 하기 땀)

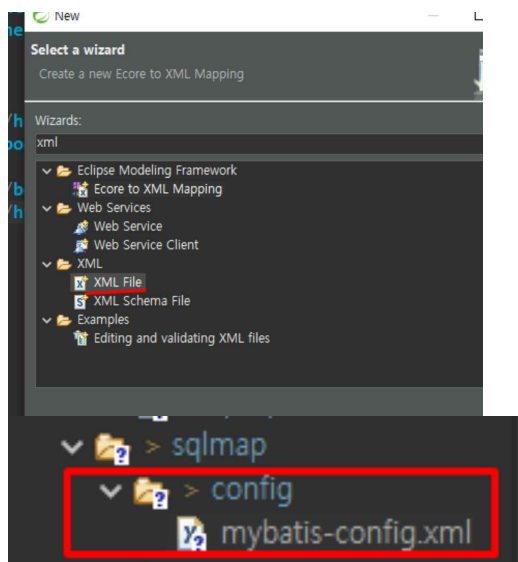
-- DEPT.XML 의 ID=deptList
SELECT * FROM DEPT;

-- Emp.xml의 id=empList
-- ENAME JOB 이 0일 수도 있는데 DEPTNO만 입력하면 SQL 문이 완성 안됨
-- 그래서 WHERE 1=1
-- 부분적으로 실행 가능 한거만 출력
SELECT * FROM EMP WHERE 1=1
AND ENAME LIKE '%' || 'A' || '%'
AND JOB LIKE '%' || 'E' || '%'
AND DEPTNO=20;
```

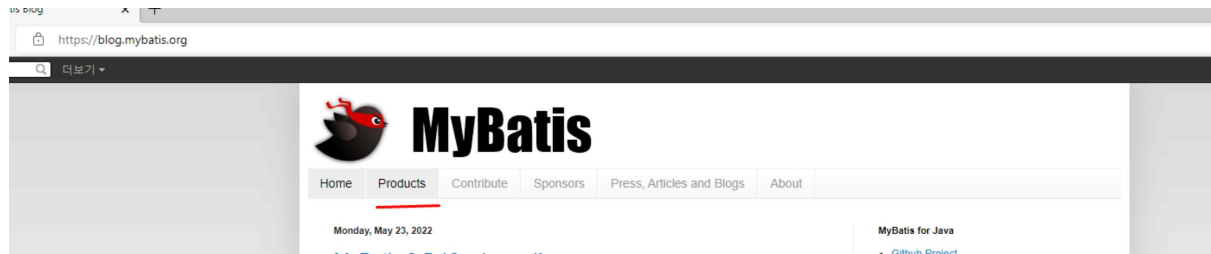
6. DTO 생성(반드시 sql 과 변수명 같게 !)



7. xml 생성 :mapper(empList,deptList) + config



https://mybatis.org/



MyBatis

Home Products Contribute Sponsors Press, Articles and Blogs About

Products

Project	Description	Links
MyBatis 3	SQL Mapping Framework for Java	download dscs

MyBatis for Java

- [Github Project](#)
- [Java Mailing List](#)

MyBatis.NET

- [.NET Google Code](#)

mybatis

Last Published: 24 May 2022 | Version: 3.5.10

REFERENCE DOCUMENTATION

- Introduction
- Getting Started
- Configuration XML**
- properties
- settings
- typeAliases
- typeHandlers
- objectFactory

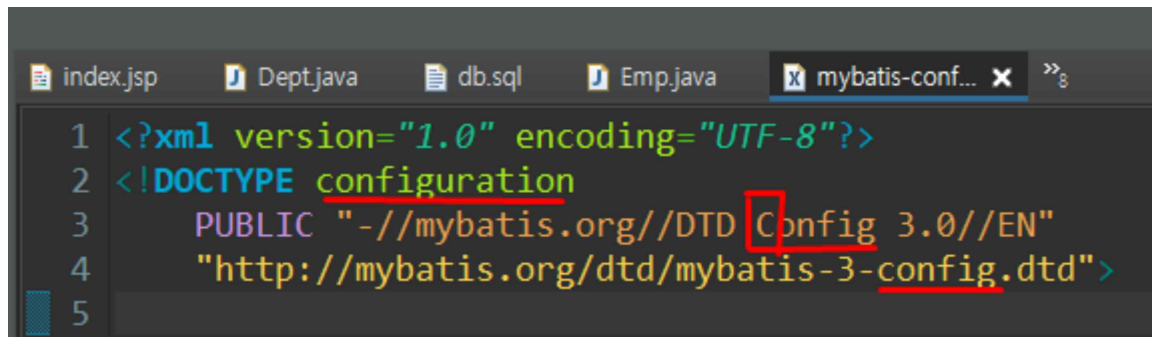
Configuration

The MyBatis configuration contains settings and properties that hav

- configuration
 - properties
 - settings
 - typeAliases
 - typeHandlers
 - objectFactory
 - plugins
 - environments
 - environment

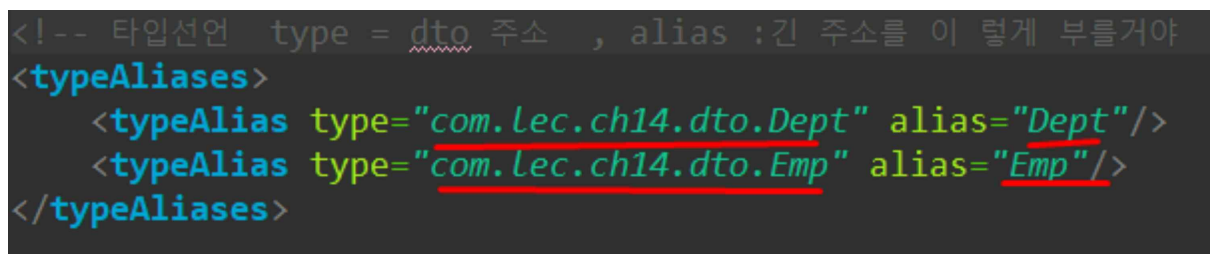
```
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
```


Mybatis-config 와서 복붙 후 수정



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE configuration
3 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4 "http://mybatis.org/dtd/mybatis-3-config.dtd">
5
```

-타입 선언

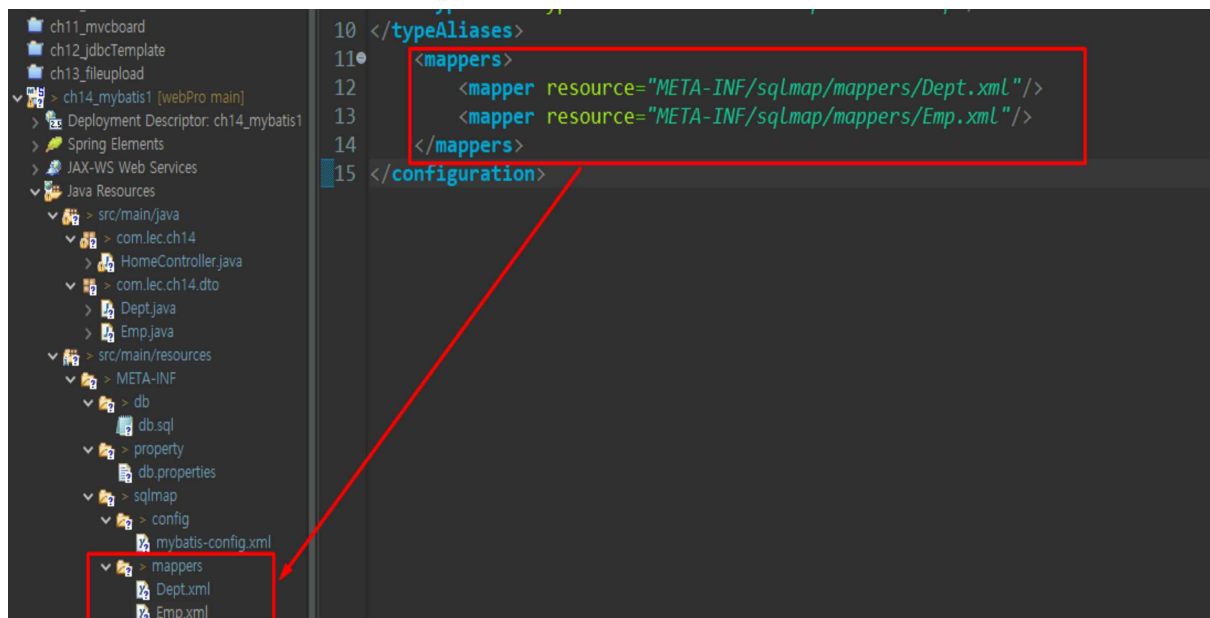


```
<!-- 타입선언 type = dto 주소 , alias : 긴 주소를 이렇게 부를거야
<typeAliases>
  <typeAlias type="com.lec.ch14.dto.Dept" alias="Dept"/>
  <typeAlias type="com.lec.ch14.dto.Emp" alias="Emp"/>
</typeAliases>
```

Type : dto 주소

Alias : 이 주소를 이렇게 간략하게 부를거야

Mappers 밑에 xml 파일 생성



```
10 </typeAliases>
11 <mappers>
12   <mapper resource="META-INF/sqlmap/mappers/Dept.xml"/>
13   <mapper resource="META-INF/sqlmap/mappers/Emp.xml"/>
14 </mappers>
15 </configuration>
```

Dept.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="Dept"> <!-- namespace 엘리아스 -->
  <!-- type : 엘리아스 이름 Result 결과가 어레이 리스트로 담을때!!!-->
  <resultMap type="Dept" id="DeptResult">
    <result property="deptno" column="deptno"/>
    <result property="dname" column="dname"/>
    <result property="loc" column="loc"/>
  </resultMap>

  <!-- select . insert . update -->
  <select id="deptList" resultMap="DeptResult"> <!-- resultMap 리턴 타입 위에 있는 id 값 -->
    SELECT * FROM DEPT
  </select>

</mapper>
```

<resultMap type = 원래는 주소 다적는건데 엘리아스 사용하여 간단하게 입력>

<select, insert , update>

Id 는 deptList 이고 리턴 타입은 위에서적은 id 값 리턴

-sql 실행 안될 때 ?

```
<![CDATA[
SELECT * FROM DEPT
]]>
</select>
```

Emp.xml

<!DOCTYPE

mapper

```
<?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"> <!-- 애 이트으 Emp 이고 type Emp 넣어 -->
  <resultMap type="Emp" id="EmpResult">
    <result property="empno" column="empno"/>
    <result property="ename" column="ename"/>
    <result property="job" column="job"/>
    <result property="mgr" column="mgr"/>
    <result property="hiredate" column="hiredate"/>
    <result property="sal" column="sal"/>
    <result property="com" column="com"/>
    <result property="deptno" column="deptno"/>
  </resultMap>
  <!-- ? 가 들어갈 예정이면 parameterType : input 받는거 -->
  <select id="empList" parameterType="Emp" resultMap="EmpResult">
    <!-- 1. 애는 무조건 실행 -->
    SELECT * FROM EMP WHERE 1=1

    <!-- ename이 널이 아니고 빈스트링이 아니면 밑에 실행해 -->
    <!-- 'A' : Emp안에있는 ename 들어간거고 -->
    <if test="ename != null and ename != ''">
      AND ENAME LIKE '%' || #{ename} || '%'
    </if>

    <if test="job != null and job != ''">
      AND JOB LIKE '%' || #{job} || '%'
    </if>

    <if test="deptno != 0">
      AND DEPTNO=#{deptno}
    </if>
  </select>
```

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

-Root-context.xml

sqlSessionFactoryBean (sql저장 장소) dataSource 의존

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE configuration
3     PUBLIC "-//mybatis.org/DTD Config 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <configuration>
6     <!-- 타입선언 type = dto 주소, alias : 긴 주소를 이렇게 불러와 -->
7     <typeAliases>
8         <typeAlias type="com.lec.ch14.dto.Dept" alias="Dept"/>
9         <typeAlias type="com.lec.ch14.dto.Emp" alias="Emp"/>
10    </typeAliases>
11    <mappers>
12        <mapper resource="META-INF/sqlmap/mappers/Dept.xml"/>
13        <mapper resource="META-INF/sqlmap/mappers/Emp.xml"/>
14    </mappers>
15 </configuration>
```

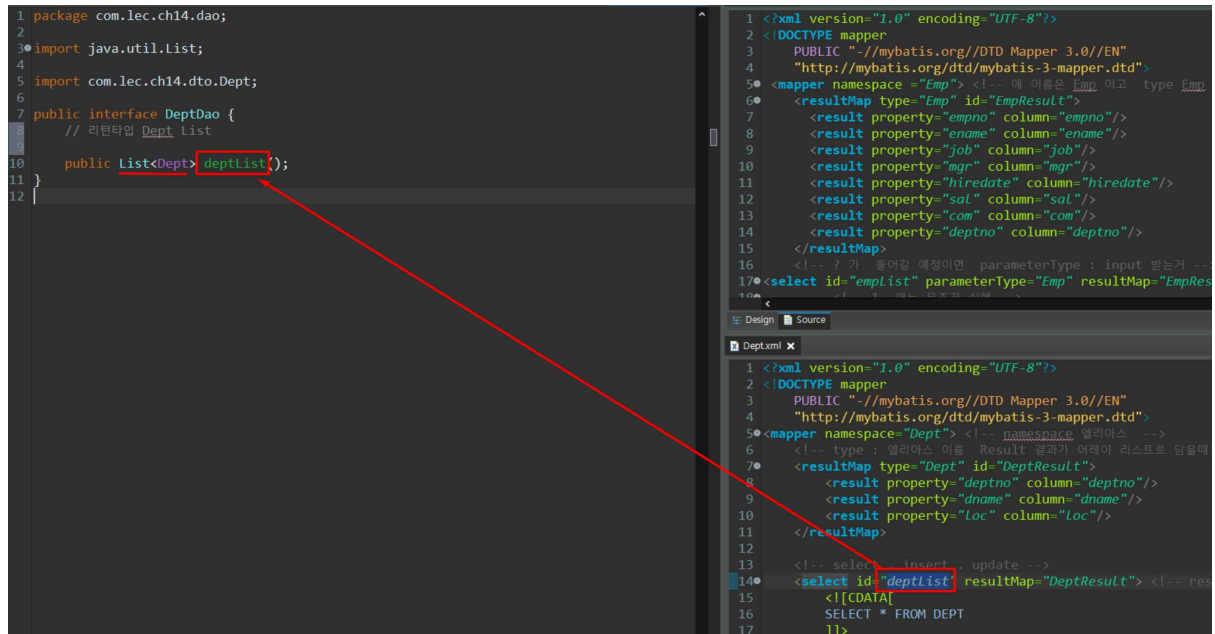
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
6     http://www.springframework.org/schema/context http://www.springframework.org/schema/context">
7
8     <!-- Root Context: defines shared resources visible to all other web components -->
9     <context:property-placeholder
10         location="classpath:META-INF/property/db.properties"/>
11
12     <!-- 필요한 빈 -->
13     <!-- 1. dataSource 데이터 연결 -->
14     <bean id="dataSource"
15         class="org.springframework.jdbc.datasource.DriverManagerDataSource">
16         <property name="driverClassName" value="${db.driverClassName}"/>
17         <property name="url" value="${db.url}"/>
18         <property name="username" value="${db.username}"/>
19         <property name="password" value="${db.password}"/>
20     </bean>
21     <!-- dataSource 의존 ! sqlSessionFactoryBean sql 저장 -->
22     <bean id="sqlSessionFactory"
23         class="org.mybatis.spring.SqlSessionFactoryBean">
24         <property name="dataSource" ref="dataSource"/>
25         <property name="configLocation" value="classpath:META-INF/sqlmap/config/
26     </bean>
```

-SeccionTemplate (sqlSessionFactory 의존)

```
<!-- dataSource 의존 ! sqlSessionFactoryBean sql 저장 -->
<bean id="sqlSessionFactory"
    class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="classpath:META-INF/sqlmap/config/mybatis
</bean>

<!-- FactoryBean 의존 (a 불러와 b 불러와) sqlSessionTemplate -->
<bean id="seccionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg index="0" ref="sqlSessionFactory"/>
</bean>
```

-DeptDao 인터페이스 생성



Public 리턴 타입은 Dept에List (java.util.List)

DeptList id 값과 같게

-implements DeptDao 받은 DeptDaoImpl 생성

EmpDao

```
1 package com.lec.ch14.dao;
2 import java.util.List;
3 import com.lec.ch14.dto.Emp;
4
5 public interface EmpDao {
6     public List<Emp> empList(Emp schEmp);
7 }
8
```

```
4 "http://mybatis.org/dtd/mybatis-3-ma
5 <mapper namespace="Emp"> <!-- 에 이름은
6 <resultMap type="Emp" id="EmpResult"
7 <result property="empno" column="e
8 <result property="ename" column="e
9 <result property="job" column="job
10 <result property="mgr" column="mgr
11 <result property="hiredate" column
12 <result property="sal" column="sal
13 <result property="com" column="com
14 <result property="deptno" column="
15 </resultMap>
16 <!-- ? 가 들어갈 예정이면 parameterTyp
17 <select id="empList" parameterType="Emp"
18 <!-- 1. 매는 무조건 실행 -->
19 SELECT * FROM E
20 <!-- ename이 널이 아니고 빈스트럼이 아니면
```

<Emp>dto

Emp 는 parameterType 에서의 EMP

-EmpDaoImpl

```
1 package com.lec.ch14.dao;
2
3 import java.util.List;
4
5 import org.apache.ibatis.session.SqlSession;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Repository;
8
9 import com.lec.ch14.dto.Emp;
10 @Repository
11 public class EmpDaoImpl implements EmpDao {
12     @Autowired
13     private SqlSession sessionTemplate;
14     @Override
15     public List<Emp> empList(Emp schEmp) {
16
17         return null;
18     }
19 }

```

```
20 <property name="password" value
21 </bean>
22
23 <!-- dataSource 의존 ! sqlSessio
24 <bean id="sqlSessionFactory"
25     class="org.mybatis.spring.S
26     <property name="dataSource"
27     <property name="configLocat
28 </bean>
29
30 <!-- FactoryBean 의존 (3. 불려와 b
31 <bean id="sessionTemplate" clas
32     <constructor-arg index="0" re
33 </bean>
34
35
36 </beans>
37
```

-EmpService 인터페이스 생성

```
1 package com.lec.ch14.service;
2
3 import java.util.List;
4
5 import com.lec.ch14.dto.Dept;
6 import com.lec.ch14.dto.Emp;
7
8 public interface EmpService {
9     public List<Dept> deptList();
10    public List<Emp> empList(Emp schEmp);
11 }
12
```

Dept 에 deptList

Emp 에 empList 함수 호출

- implements 받는 EmpServiceImpl 생성

```
@Service
public class EmpServiceImpl implements EmpService {
    @Autowired
    private DeptDao deptDao;
    private EmpDao empDao;

    @Override
    public List<Dept> deptList() {
        return deptDao.deptList();
    }

    @Override
    public List<Emp> empList(Emp schEmp) {
        if(schEmp.getEname() != null) { // 이름이 널이아니면
            schEmp.setEname(schEmp.getEname().toUpperCase()); // 대문자로 변환
        }
        if(schEmp.getJob() != null) {
            schEmp.setJob(schEmp.getJob().toUpperCase()); // 대문자로 변환
        }
        return empDao.empList(schEmp);
    }
}
```


-EmpController 생성

```
@Controller
public class EmpController {
    @Autowired
    private EmpService empService;
    @RequestMapping(value="emp", method = RequestMethod.GET)
    public String emp(@ModelAttribute("schEmp")Emp schEmp, Model model) {
        model.addAttribute("empList", empService.empList(schEmp));
        model.addAttribute("deptList", empService.deptList());
        return "emp";
    }
}
```

-Model 이용해 empList , deptList 연결

Return "emp.jsp" 로

