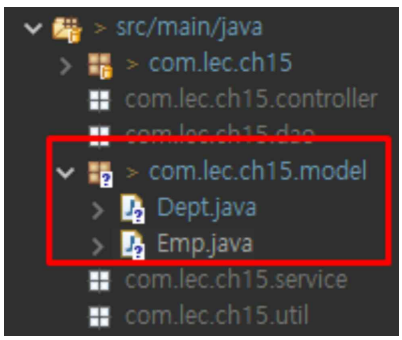
## Pom.xml 이번만 예전 버전 (try-catch위해)

## Wep .xml 한글처리

```
<!-- JDBC template -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-jdbc</artifactId>
			<version>4.1.4.RELEASE</version>
		</dependency>
		<!-- myBatis -->
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis</artifactId>
			<version>3.2.8</version>
		</dependency>
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis-spring</artifactId>
			<version>1.2.2</version>
		</dependency>
```

## Dto 생성



Emp.dto

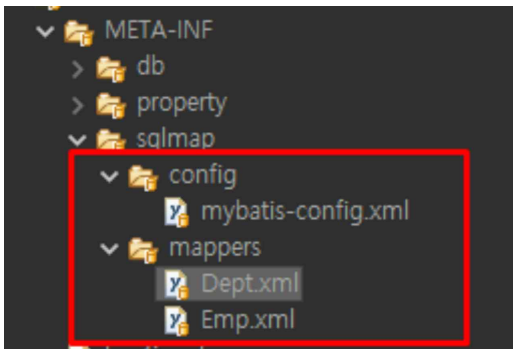
```java
import lombok.Data;
@Data
public class Emp {
    private int empno;
    private String ename;
    private String job;
    private int mgr;
    private Timestamp hiredate; //
    private int sal;
    private int comm;
    private int deptno; // join
    private String loc; // join
    private int startRow; //페이징 처리를 위한 속성 변수
    private int endRow; // 페이징 처리를 위한 속성 변수

}
```

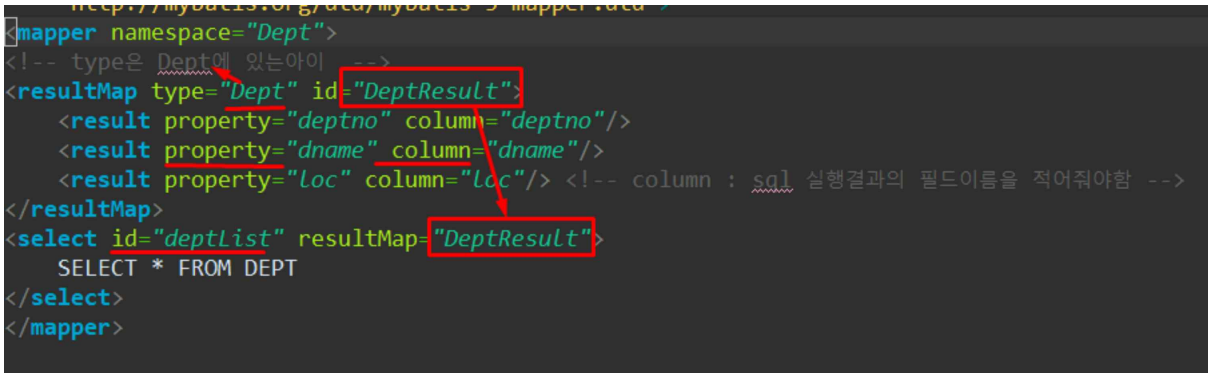
Dept 테이블 조인

페이징 처리

Config(Dept.xml , Emp.xml) 생성

dataSource
mybatis-config.xml






Dept.xml

Type은 Dept에 있는아이

Id 값은 : DeptResult

Property : dto에 있는 속성 값

Column : sql 실행 결과의 필드이름을 적어줘야함

Select id : sql id 값 셋팅

resultMap = 리턴값은 위에 id 값 DeptResult

## Emp.xml

```xml
<mapper namespace="Emp">
    <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="comm" column="comm"/>
     <result property="deptno" column="deptno"/> <!-- Dept테이블 join -->
     <result property="dname" column="dname"/> <!-- Dept테이블 join -->
     <result property="loc" column="loc"/>        <!--Dept테이블 join -->
    </resultMap>
    <select id="empList" parameterType="Emp" resultMap="EmpResult">
        SELECT * FROM (SELECT ROWNUM RN, A.* FROM(SELECT * FROM EMP ORDER BY EMPNO) A)
        WHERE RN BETWEEN #{startRoe} AND #{endRow}
    </select>
</mapper>
```

**<select Id : empList**

**parameterType은 "Emp"**

**리턴값 EmpResult**

## ArrayList 는 EmpResult 사용 !!

## -totCnt

```xml
<select id="totCnt" resultType="int"> <!-- id 는 totCnt  int 값 리턴 이여서 resultType = "int" -->
SELECT COUNT (*) FROM EMP
</select>
```

**Id는 totCnt**

**resultType : int  sql 결과가 숫자(int) 이기 때문에 "int" 사용**

-detail (상세보기)

```
<!-- 한줄만 입력 받는아이는 resultType  EMPNO=7698로 상세보기여서 type은 int  -->
<select id="detail" parameterType="int" resultType="Emp">
    SELECT * FROM EMP WHERE EMPNO=#{empno}
</select>
```

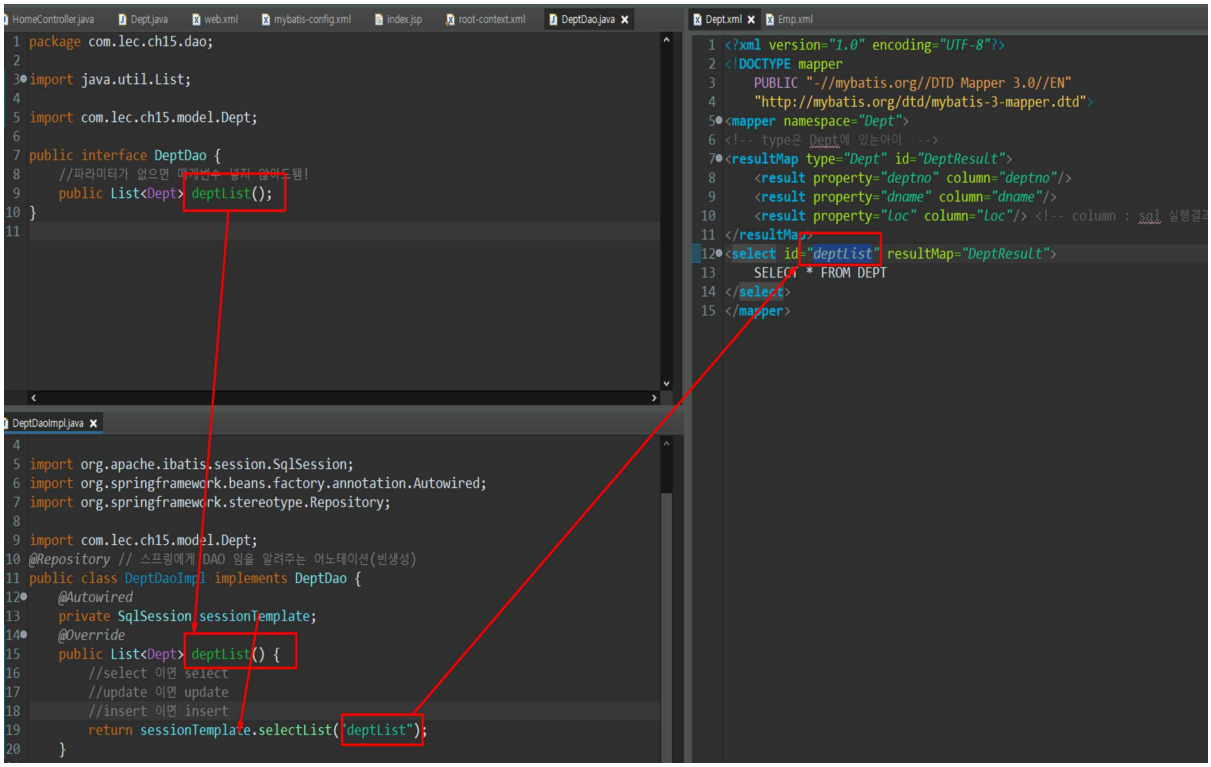
Id 는 detail

parameterType = “int” 인데 이유는 empno가 숫자이기 때문에

resultType = “Emp” (받는게 list 아닐때 resultType사용)

## - root-context.xml 2번째 3번째 처리

DeptDao 인터페이스생성 (dao 공부할때 확인)



Dao 요약 들어 갈 곳

EmpDao에서 xml에 입력한 sql 불러오기

-empList

```
<select id="empList" parameterType="Emp" resultMap="EmpResult">
        SELECT * FROM (SELECT ROWNUM RN, A.* FROM(SELECT * FROM EMP ORDER BY EMPNO) A)
WHERE RN BETWEEN #{startRow} AND #{endRow}
</select>
```

```
public List<Emp> empList(Emp emp);
```

xml 리스트는 resultMap - > dao List< >

parameterType = “Emp” - > dao (Emp emp)

-detail

```
<!-- 한줄만 입력 받는아이는 resultType  EMPNO=7698로 상세보기여서 type은 int  --
<select id="detail" parameterType="int" resultType="Emp">
        SELECT * FROM EMP WHERE EMPNO=#{empno}
</select>
```

```
public Emp detail(int empno); // resultType : Emp , parameterTyp
```

xml : 한줄만 나오는아이는 (resultType=(Emp타입))

-> dao : Emp(resultMap에 type)

parameterType = “EMPNO가 숫자이기 때문에 int #empno “

- > dao : int empno

## -managerList

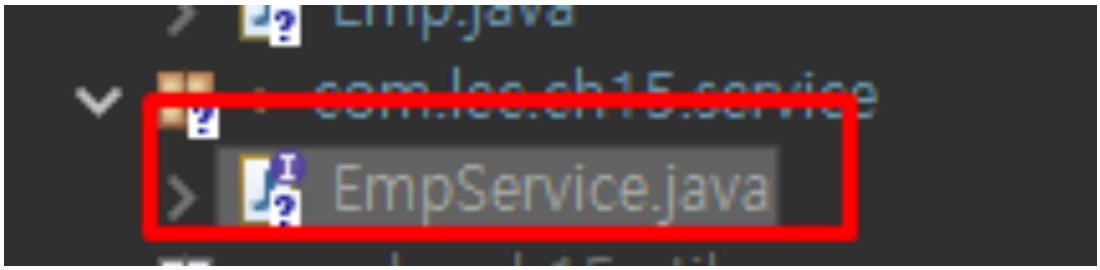
```
<select id="managerList" resultMap="EmpResult">
        SELECT * FROM EMP WHERE EMPNO IN (SELECT MGR FROM EMP)
</select>
```

애는 type ?

```
public List<Emp> managerList(); // 파라미터 타입 없음
```

- 리스트 여서 resultMap 사용 애는 왜 파라미터 타입이 없을까 ?

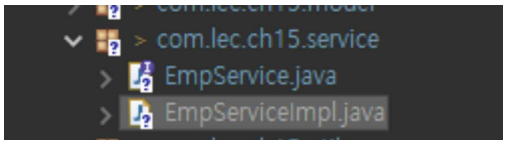
**EmpService 인터페이스 생성**




**-Paging.java 클래스 만들기 com.lec.util**

```
package com.lec.ch15.util;

import lombok.Data;

@Data
public class Paging {
    private int currentPage = 1;
    private int pageSize;
    private int blockSize;
    private int startRow;
    private int endRow;
    private int totCnt;
    private int pageCnt;
    private int startPage;
    private int endPage;
    // Paging pagine = new Pageing(14, pageNum, 10 , 5)
    // Paging pagine = new Pageing(14, pageNum)
}
```

## -EmpService implements 받은 EMpServiceImpl 생성




## -EmpController

```java
import com.lec.ch15.service.EmpService;
import com.lec.ch15.util.Paging;

@Controller
public class EmpController {
    @Autowired
    private EmpService empService;
    @RequestMapping(value="empList", method=RequestMethod.GET)
        public String empList(String pageNum, Model model) {
            // empList.do?pageNum = 2
            model.addAttribute("empList", empService.empList(pageNum));
            model.addAttribute("paging", new Paging(empService.totCnt(), pageNum, 10, 5));
            return "empList";
        }

}
```

```java
public class EmpServiceImpl implements EmpService {
    @Autowired
    private DeptDao deptDao;
    @Autowired
    private EmpDao empDao;
    @Override
    public List<Dept> deptList() {
        // TODO Auto-generated method stub
        return deptDao.deptList();
    }

    @Override
    public List<Emp> empList(String pageNum) {
        Paging paging = new Paging(empDao.totCnt(), pageNum, 10,
        Emp emp = new Emp();
        emp.setStartRow(paging.getStartRow());
        emp.setEndRow(paging.getEndRow());
        return empDao.empList(emp);
    }

```

## empList.jsp

## 등록 실패 후 입력 했던거 안날라가게 하는법

### write.jsp

```
        var empno = frm.empno.value; // 너가친 사번
        var ename = frm.ename.value;
        var job = frm.job.value;
        var mgr = frm.mgr.value;
        var hiredate = frm.hiredate.value;
        if(! hiredate){ // 날짜 입력 안하면 오늘 로 해라 !!
            hiredate = '<%=new Date(System.currentTimeMillis())%>';
        }
        var sal = frm.sal.value;
        if( ! sal){ // 급여 입력 안하면 0으로 해라 !!@!!!!
            sal = 0;
        }
        var comm = frm.comm.value;
        if(! comm){
            comm = 0;
        }
        var deptno = frm.deptno.value;
        location.href = 'confirmNo.do?empno='+empno+'&ename='+ename +'&job='+job +
                '&mgr='+ mgr + '&hiredate='+ hiredate + '&sal='+sal + '&comm=' + comm
                '&deptno=' + deptno;
    }
```

-Write.do에 입력한 값 넘 겨 주고

```
    }
    // 10. 등록하기 ok처리
    @RequestMapping(value = "write", method = RequestMethod.POST)
    public String write(Emp emp,  Model model) { // model이 emp 값 들어있음
        try {
            model.addAttribute("writeResult", empService.insert(emp));
        } catch (Exception e) {
            model.addAttribute("writeResult", "필드 값이 너무 깁니다. 등록 실패 ")
            return "forward:writeView.do";
        }
          return "forward:empDeptList.do";
    }
```

-모든 값  Emp emp 로 받음

- 등록 실패 면 다시 뷰 단으로 넘어 갈 때 입력받은 값 emp 를 가지고 다시 간다

```
<th>사번</th>
<td>
    <input type="number" name="empno" required="required" value="${emp.empno}">
    <input type="button" value="중복확인" onclick="chk()" class="btn"><br>
    <span><c:if test="${not empty msg }">${msg }</c:if></span>
</td>
</tr>
<tr>
  <th>이름</th>
  <td>
      <input type="text" name="ename" value="${emp.ename }">
  </td>
</tr>
<tr>
  <th>직책</th>
  <td>
      <input type="text" name="job" value="${emp.job }">
```

Value 를 이용해 입력 받은 값을 받아 온다

```
<c:if test="${not empty writeResult }">
    <script>alert('${writeResult}')</script>
</c:if>
```

실패 후 다시 돌아 올 때 alert 메시지 뿌려주고