에이콘 아카데미

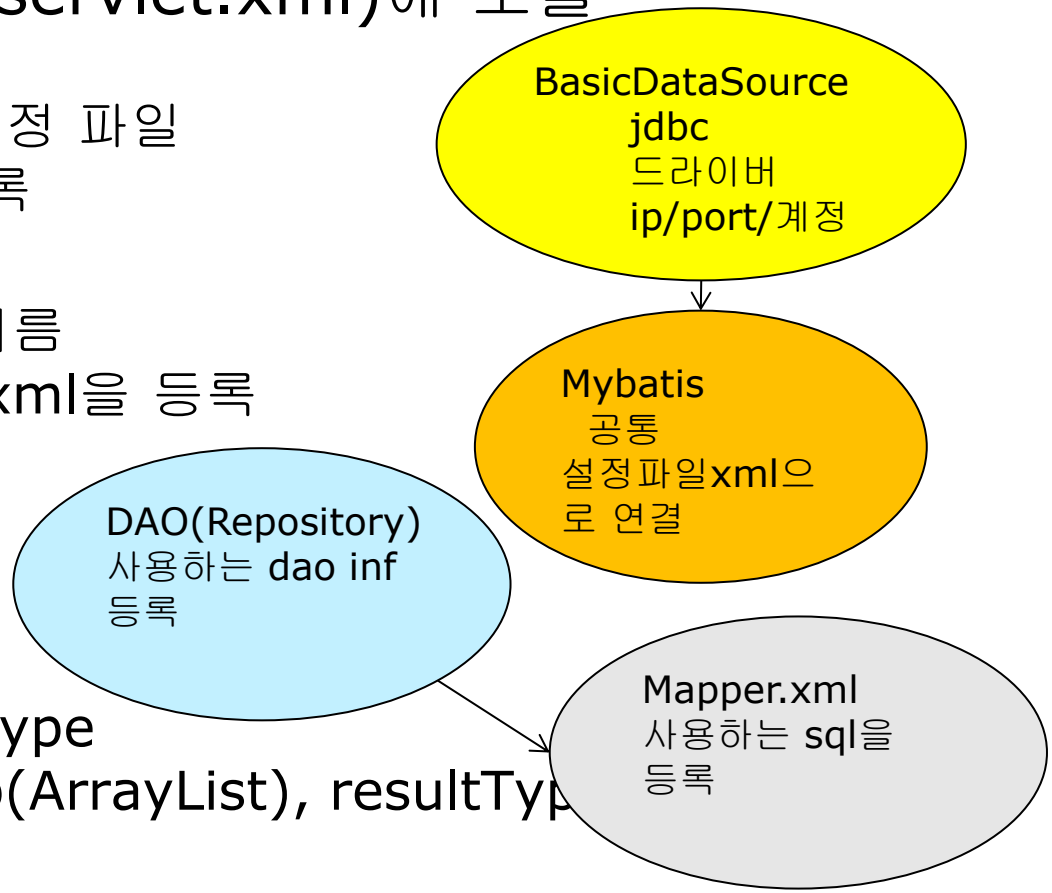
Mybatis

zelratole@hanmail.net

Contents

Mybatis연결

- 컨테이너(dispatcher-servlet.xml)에 모듈
  - DB 연결정보
  - mybatis 모듈 등록, 설정 파일
  - dao interface 위치 등록
- Mybatis 설정
  - 공통 vo를 등록, alias이름
  - 업무별 XXXXmapper.xml을 등록
- Mapper.xml
  - namespace 설정
  - resultMap 선언..: vo
  - sql 선언 :select
    - 입력값 :paramterType
    - return : resultMap(ArrayList), resultTyp
    - update, insert



dispatch-servlet.xml(조립기)

- 연결:*BasicDataSource : dataSource*
  - 드라이버
  - DB서버 정보(ip, port, sid, 계정, 비밀번호)
- mybatis 설정
  - 연결 정보를 받아서 mybatis와 설정..
    - dataSource ref => *dataSource*
  - mybatis 공통 설정 config xml파일 선언
    - configLocation
- Repository(DAO) 설정
  - *MapperScannerConfigurer를 통해서 dao단의 패키지를 선언.*

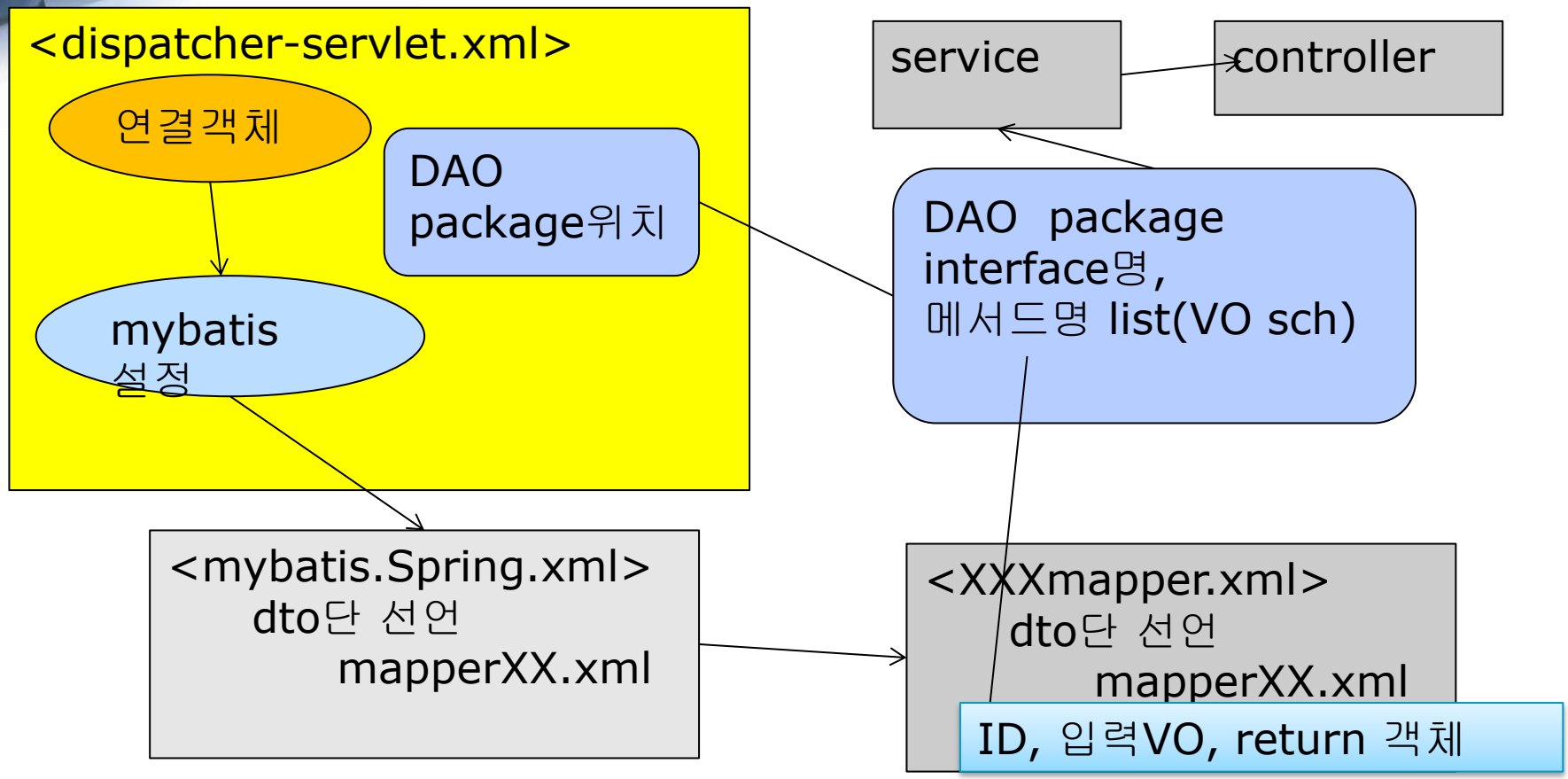
mybatis 공통 설정 config xml

- *classpath:resource/mybatis.Spring.xml*
  - classpath:src를 기준으로 잡힘.
  - resource폴드에 mybatis공통 설정 파일인 mybatis.Spring.xml
- 주로 하는 역할
  - 모듈별로 실제 처리할 sql 파일(xml)을 등록처리
    - <mapper resource="EmpMapper.xml"/>
  - 공통 VO, DTO등을 alias으로 지정, sql파일에서 활용성을 높이기 위해, 간단한 이름으로 설정.
    - *springweb.z01_dto.Emp ==> emp*

XXXMapper.xml

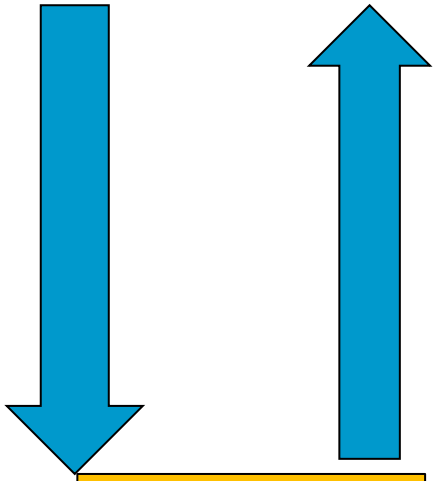
- mybatis 설정 config.xml 에서 모듈별로 sql을 사용할 수 있게끔 처리하는 파일
  - sql 처리 :select * from
  - 인식할 수 있는 id명 : dao단에서 호출 시 필요
    - hello.CallDao.callList
  - 입력관련 데이터 값 처리 parameType="CallDto"
  - return값에 대한 처리 : resultMap =VO가 모인 ArrayList형태
  - namespace : hello.CallDao를 선언하면 id에 namespace명 생략가능
  - DAO단 interface와 밀접한 연관관계
  - dao단 interface 패키지명.인터페이스명.메서드명
    - package hello;
    - public interface CallDao{
    - public ArrayList<Vo> callList(CallDto dto);

호출 상관관계도


<dispatcher-servlet.xml>
연결객체
DAO
package위치
mybatis
설정
service
controller
DAO package
interface명,
메서드명 list(VO sch)
<mybatis.Spring.xml>
dto단 선언
mapperXX.xml
<XXXmapper.xml>
dto단 선언
mapperXX.xml
ID, 입력VO, return 객체

mybatis MVC 패턴

- 화면단(*.jsp)
- Controller
- Service
- Repository(Dao)
- mybatis SQL (*.xml)



Repository
(DAO)

XXXmapper.
xml

Controller
요청, 모델
view결정

Service
비지니스로직
Model

workflow

외부모듈연계

View
(*.jsp)

개발 process

- 화면단 구성(.jsp)-css, img, html...
- sql 처리(raw data 확인)
  - ex) select empno, ename
    - from emp
    - where ename like '%'||'HIMAN'||'%'
    - and job like '%'||'SUPERMAN'||'%'
- DTO, VO, ArrayList<DTO>
  - class Emp{
    - private int empno;
    - private String ename;
    - private String job;

개발 process

- Repository(DAO)
  - public interface Emp_Dao
  - // return될 값 : ArrayList<Emp>
  - // 조건처리할 값 : Emp sch
    - public ArrayList<Emp> empList(Emp sch);
- mybatis.Spring.xml
  - 핵심 DTO/VO 클래스를 alias이름으로 등록
    - <typeAlias alias="emp" type="springweb.dto.Emp"/>
  - sql을 처리할 XXXMapper.xml을 등록
    - <mapper resource="resource/EmpMapper.xml"/>

개발 process

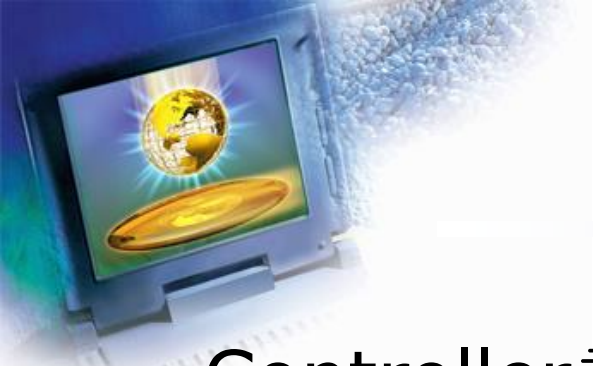
- EmpMapper.xml 처리(sql)
  - repository/dao를 연결할 id
    - namespace로 선언
      - dao패지명.인터페이스명
    - id : dao인터페이스의 메서드명
  - 입력처리 할 dto/vo단 처리
    - <select parameterType="emp"
      - where ename like '%'|| #{ename} ||'%'
        - ==> emp.getEname()
  - return할 객체(dto, ArrayList<vo>)
    - <select id="메서드"

```
select empno, ename
    from emp
    where ename like
'%'||'HIMAN'||'%'
    and job like
'%'||'SUPERMAN'||'
%'
```

개발 process

- EmpMapper.xml 처리(sql)
  - return할 객체(dto, ArrayList<vo
    - resultMap : VO --> ArrayList
      - <resultMap type="한개의 VO객
      - <result column="ename" property="name" />
        - select된 컬럼명은 ename인 것을 setName()으로 할당 및, getName()으로 호출처리된다.
    - resultType : VO
    - <select resultMap="empMap"
- Sevice 단에 처리
  - DAO단 autowire를 통해서 처리..
  - 조건, 반복문 처리...핵심 Model 데이터 처리..

```
select empno, ename
  from emp
  where ename lik
'%'||'HIMAN'||'%'
    and job like
'%'||'SUPERMAN'||
%'
```



개발 process

- Controller처리
  - serice호출
  - model 처리
  - view 선택
- 화면단 마무리
  - 넘겨온 model 데이터 el/jstl 처리
  - javascript, jquery 처리..

수정 처리

- 리스트 된 데이터 클릭 시, 상세화면으로 전환 처리
  - 초기 화면구성
  - sql 작성
  - DAO 한개 데이터 처리 메서드 추가
  - DB.xml 처리..
  - service단
  - controller 모델처리
  - view단 :jstl, el 처리..

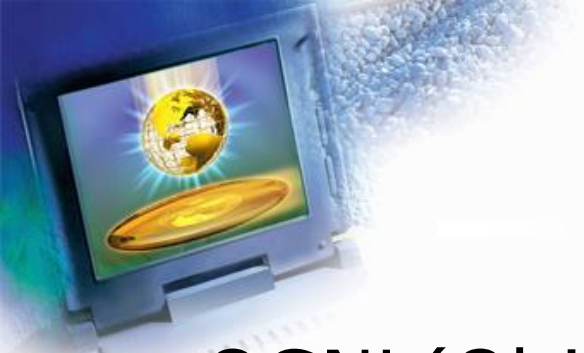
수정처리확인예제

- http://localhost:6080/springweb/deptlist_my.do를 기준으로 수정 처리
- 수정 처리
  - sql 작성
  - VO/DTO
  - DAO 수정 처리 메서드 추가 ...
  - xml
  - service
  - controller
  - 화면단 정리

- select * from DEPT 결과물을 처리하는 DAO 만들기
- VO : 단위 데이터를 저장할 객체
  - Dept
- DAO단(Repository)
  - interface DeptDao
  - 메서드명 detpList
    - public ArrayList<Dept> deptList();
- mybatis.Spring.xml
  - Dept의 alias이름 설정
  - DeptMapper.xml 생성 및 선언
- DeptMapper.xml
  - namespace 선언
  - <resultMap id="deptRsMap" type=""
  - <select id="" resultMap=""

DAO 만들기 연습

- select * from member로 데이터 가져오기
- DTO
- DAO단
  - 메서드선언
- mybatis.Spring.xml 처리
- memberMapper.xml 처리



XML에서 동적 SQL 처리

- OGNL(Object Graph Navigation Language)
  - ex) <c:if..., <c:forEach ...
  - 프로퍼티의 값을 가져오거나 설정하기 위한 언어..
  - 웹워크, JSTL 등에서 활용, mybatis
- 마이바티스가 제공하는 XML 엘리먼트
  - if : 조건처리
  - choose(when, otherwise) : 조건 그외 처리..
  - trim(where, set)
  - foreach : 반복문

if 엘리먼트

- 모든 조건에 대해 처리하고 만족하는 조건의 결과를 모두 적용한다.
  - <select id="selData" parameterType="emp">
    - SELECT * FROM emp
    - <if test = "ename != null">
    - WHERE ename = #{ename}
    - </if>
- 동적query : 위와 같이 sql구문이 조건에 따라서 변경되는 것을 말한다.
- 여러 논리 조건
  - <if test="ename !=null and sal !=0">

기본예제

- emp 리스트 데이터 중, ename이나 job 등 해당 데이터가 있을 때, 검색이 되고, 그렇지 않으면 전체 검색이 되는 동적 query문을 작성하세요..(조건으로 ename만, ename과 job이 있을 때, job만 있을 때, 전체 조건이 없을 때)
  - SELECT *
  - FROM EMP
  - WHERE 1=1
  - <if test="ename!=null">AND ename=#{ename}</if>
  - <if test="job!=null">AND job=#{job}</if>

choose(when, otherwise)

- 조건 지정과 그 외에 대한 처리를 동적인 sql에서 진행할 때 활용된다.
  - <SELECT parameterType="emp" >
    - SELECT * FROM EMP WHERE 1=1
    - <choose>
      - <when test="sal>5000">AND deptno=10</when>
      - <when test="sal>4000">AND deptno=20</when>
      - <when test="sal>1000">AND deptno=30</when>
      - <otherwise>AND deptno=40</otherwise></choose>

trim(where)엘리먼트

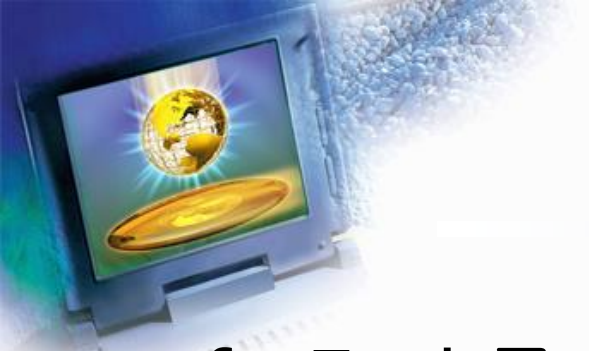
- if 엘리먼트가 조건 처리에서 where구문에 대한 처리를 보완하기 위해 사용한다.
  - prefix : 처리 후 엘리먼트 내용 있으면 앞에 붙여줌
  - prefixOverrides : 처리 후 앞에 해당 문자 있으면 자동으로 지워줌.
- select *
- from emp
- where 1=1 : 구분의 활용하지 않으면
- <where>
- <trim prefix="WHERE" prefixOverrides="AND | OR">
  - <if test="ename!=null>ename=#{ename}
  - <if test="job!=null>and job=#{job}

foreach엘리먼트

- 동적 sql을 loop문으로 반복처리 해야 할 경우
  - 조회조건에서 부서를 여러 개 검색조건으로 처리할 때..
    - select * from emp
    - where deptno in (10, 30, 20);
- 속성 : <foreach collection="" item=""
  - collection : 배열의 목록 리스트를 가져옴 ex)jstl items
  - item : 배열의 목록 단위 ex) jstl의 var
  - index : 몇 번째 값이지 나타내는 인덱스, 0
  - open : 목록에 값을 가져와 설정 할 때, 앞에 붙여 주는 문자 ex) deptno in **(**
  - close : 목록에 값을 가져와 설정 한 후, 마지막에 붙여 주는 문자. ex) deptno in (..... **)**
  - separator : 목록에 값을 가져와 설정할 때 값들 사이에 붙여 주는 문자열 ex) deptno in(10**,** 20**,** 30)

멀티 선택 처리..

- 화면
  - 검색할 부서 []10 []20 []30
    - <input type="checkbox" name="deptnos"/>10
    - <input type="checkbox" name="deptnos"/>20
    - <input type="checkbox" name="deptnos"/>30
- DTO, VO
  - private String[] deptnos;
  - setDeptnos(String[]), getDeptnos()
- forEach를 활용한 동적 sql

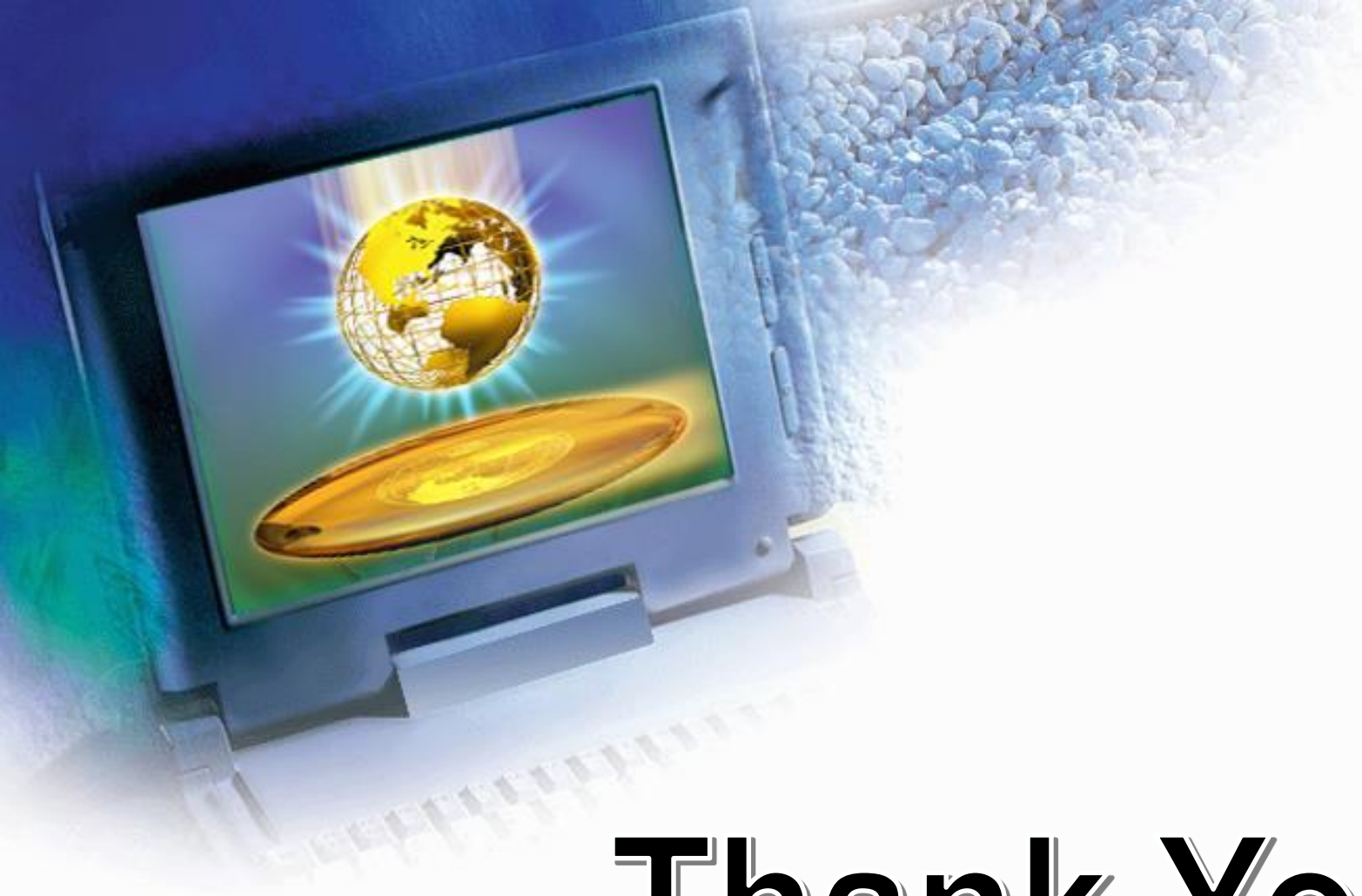


멀티 선택 처리..

- forEach를 활용한 동적 sql
  - select * from emp where 1=1
  - <if test="deptnos !=null"> ==> getDeptnos()
    - deptno in
    - <foreach collection="deptnos" item="ckdeptno" index="idx" open="(" close=")" separator=",">
      - #{ckdeptno}

멀티 선택 확인 예제

- job을 멀티선택으로 검색 처리..
  - CLERK,SALESMAN,PRISIDENT,MANAGER,ANALYST
- 화면단 : 멀티선택으로 변경
- DTO : 추가/변경
- DAO : 추가/변경



Thank You !

zelratole@hanmail.net