1일차 (04/05)

1. JSP (View) ⇒ jsp → java → class 로 변환

1) 구성요소.

① Directive element <%@ %> : 지시어 : <%@ Page ... %> → JSP 라고 알려주는 역할이기 자주사용.

② Scripting elements

㉠ Scriptlet <% %>

: Public void Service () { } : 메소드 구현

㉡ Declaration <%! %>

: Class { } : 클래스 구현

㉢ Expression <%= %>

: <%=str %> == <% out.println(str) %> 서비스에서 구현

클라이언트 요청시 발생

③ JSP 액션 == JSP 표준태그 <jsp: xxx />

④ EL (Expression Language) ${ }

⑤ JSTL (Java Standard Tag Library) <c: xxx>

cf) CTL (Custom Library)

2) Bean (POJO) JSP Bean

① Data

② Service

③ Visual Component

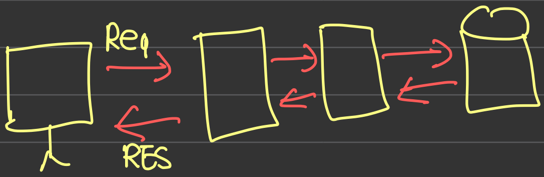
3) API

① Servlet

② JSP

③ default : Java SE

2. MVC



↳ JSP : HTML/CSS/JS — 동기

비동기: text, XML, JSON

3. Framwork

① Frontend : JQuery / Bootstrap / Angulars / React / Vue ...

② Backend : Spring (M+C) / Struts (C) ...

③ ORM : MyBatis / Hibernate ...

4. DBMS : DB관리 소프트웨어

: 오라클 / MySQL / MsSQL ...

5. 스프링 MVC Pj 구조

- main/src : Java (M/C)
- main/resource : XML (log, myBatis
- test/src : java for junit
- test/resource : xml for junit
- Lib (JDK + Spring)
- src/main/webapps/WEB-INF/views : JSP
- " /resources : CSS, JS, image, audio, Video

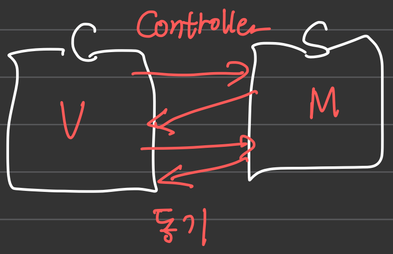
4가지 XML : Web.xml / root-Context.xml / Servlet-Contex.xml / Pom.xml

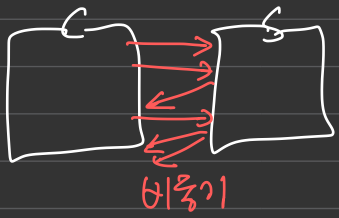
2일차

동기 vs 비동기.

① 동기: 전체 화면 갱신 (Html 문서 전체를 갱신) : JSP

② 비동기: 부분갱신(Data만전달) ex 자동완성기능 : text/xml/JSON





JAVA ≒ OS

Java SE로구성.

JDBC : Java DataBase Connectivity ≒ 핸들링

Oracle / MySQL / MsSQL ≒ 프린터1 프린터2 . . . .

드라이버제공

JSP Default Object

객체를 따로 생성 X 스크립트렛 <% %>에서 사용가능한 객체 서비스에서만 사용

① Out (type: JspWriter) :

② request (type: HttpServletRequest)

③ respone (type : HttpServletResponse)

④ Session (type : HttpSession)

⑤ application (type : Servlet Context)

⑥ page (Type : Object)

⑦ Page Context (type: PageContext)

⑧ config (type : ServletConfig)

⑨ exeption(type : Throwable)

EL

${ }

기능: 속성값을 얻어온다 / 배열의 값을 얻어온다 / List의 값을 얻어온다 / Map의 값을 얻어온다.

EL의 내장객체

4일차

- 인간의 데이터 표현

문자열(테이블+차트) + 이미지 + 소리 +영상

- OCJP : 자바자격증

- 서비스의 (메소드)

지역변수: 파라미터 지역변수 vs 선언초기화 지역변수

객체의 존재이유

멤버 / 메서드.

(특성 / 기능)

Life Cycle : 지하차고

init : 첫번째 요청 → 메모리에 올라감

JSP → servlet → class

service: 요청할때마다

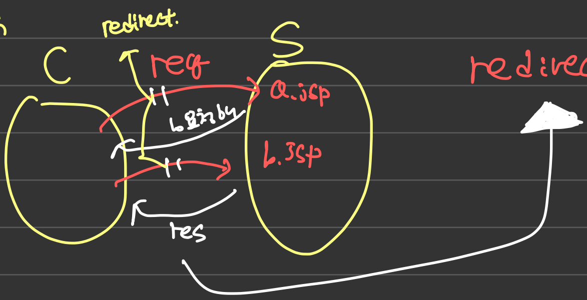
Destroy : 메모리 언로딩

DML = Insert/Delet/update

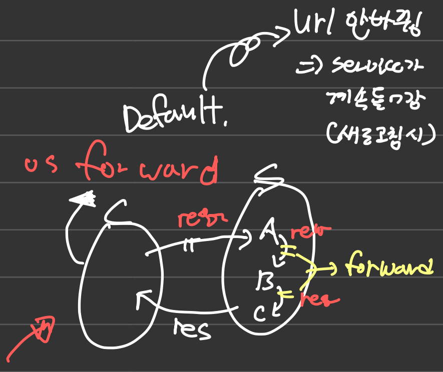
서비스 > DAO   서비스 ( ) { ①.②③ }

DAO

업무로직



redirect vs forward



Page Scope

vs

request scope 이게 넓은의미.

범위 같음

4/9

제약조건 (Contraint) : 5가지 ⟹ 무결성 보장하기 위해

- PK (유일 + Not Null)
- Unique (유일)
- FK
- Not Null
- check (필터링)

ex) 부서 (부모)	vs	사원 (자식)
부서번호 (PK) : U or P		사원번호 (PK)
부서이름		이름
부서위치		부서번호 (FK)

Pagination.

원칙 ① Get방식 처리

② 페이지 하단에 페이지 들의 번호를 보여주고, 원하는 번호를 선택하면 해당페이지 이동해서
목록을 보여줘야한다

③ 페이징은 반드시 필요한 번호만 출력해야 한다. 만약 42개 게시글 / 페이지당 10개 → 5페이지 되어야함.

④ 이전/ 다음버튼이 필요.

⑤ 게시글 조회, 수정/삭제후 원래 페이지로 이동해야함 → 5페이지 수정/조회/ 삭제경우 다시 5페이지로 이동 해야함.

MySQL Query 문

select * from 테이블명 order by 게시글번호 desc limit 시작행, 출력할 개수.

-JAVA

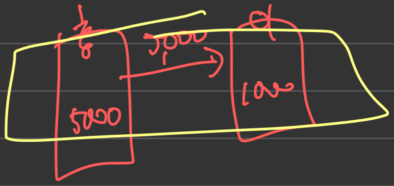
pageStart = (현재 페이지 번호 - 1) * 페이지당 보여질 게시글 수

4/13

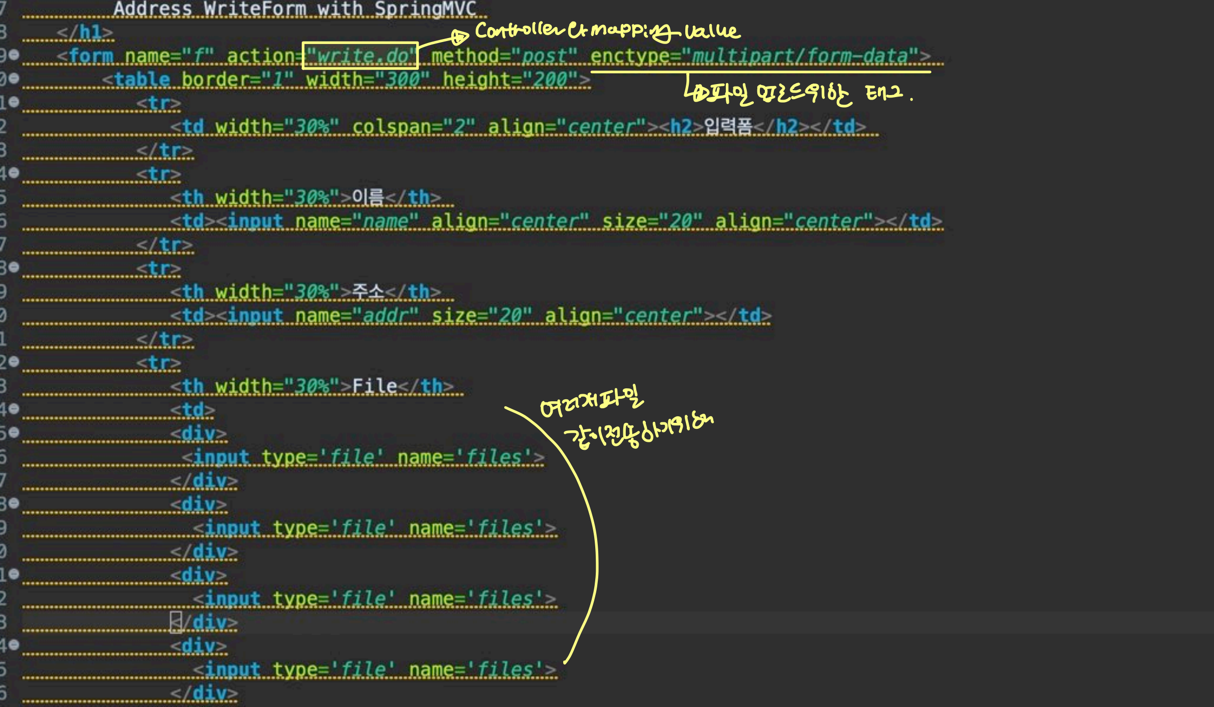
transation 개념.

작업단위.

: 떨어질수 없는 상태들의 변화를 묶는것



→ 하나라도 안됐을때 그 전의 상태유지.

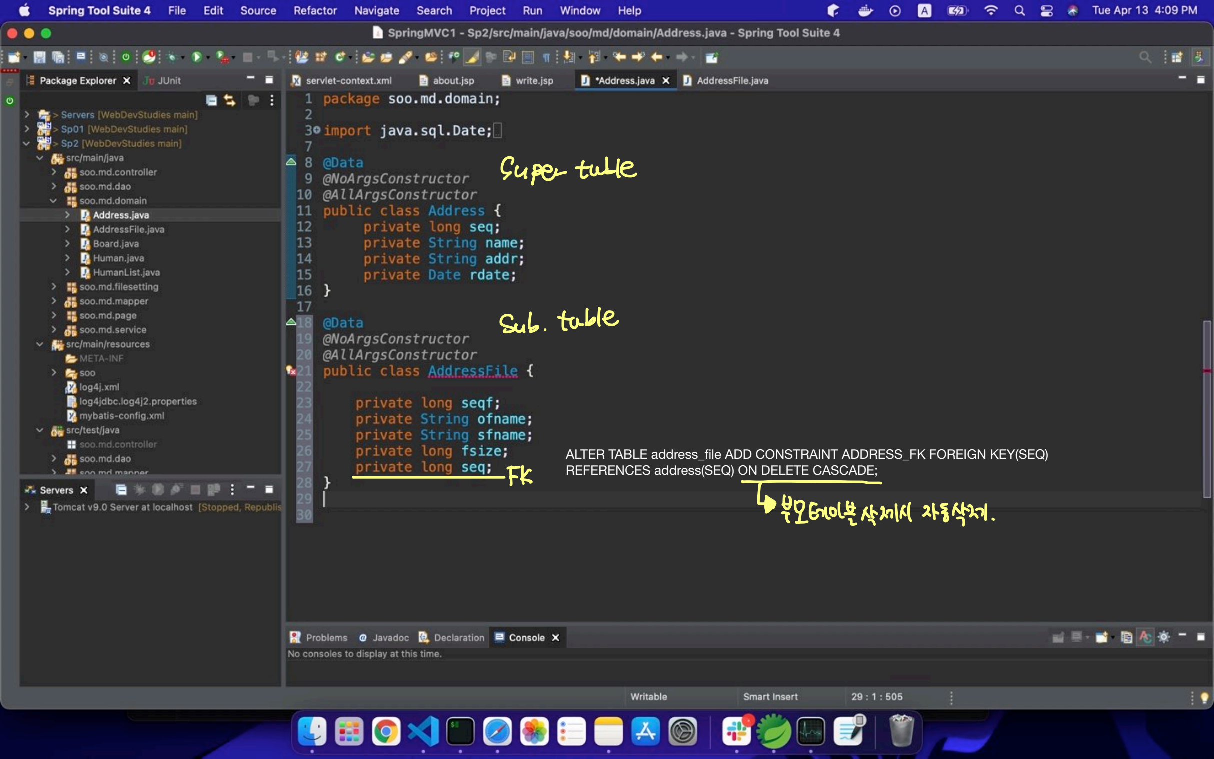


```
        Address WriteForm with SpringMVC
    </h1>
    <form name="f" action="write.do" method="post" enctype="multipart/form-data">
        <table border="1" width="300" height="200">
            <tr>
                <td width="30%" colspan="2" align="center"><h2>입력폼</h2></td>
            </tr>
            <tr>
                <th width="30%">이름</th>
                <td><input name="name" align="center" size="20" align="center"></td>
            </tr>
            <tr>
                <th width="30%">주소</th>
                <td><input name="addr" size="20" align="center"></td>
            </tr>
            <tr>
                <th width="30%">File</th>
                <td>
                <div>
                 <input type='file' name='files'>
                </div>
                <div>
                  <input type='file' name='files'>
                </div>
                <div>
                  <input type='file' name='files'>
                </div>
                <div>
                  <input type='file' name='files'>
                </div>
```

Controller단 mapping value

⇒파일 업로드위한 태그.

여러개파일
같이전송하기위해



```java
package soo.md.domain;

import java.sql.Date;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class Address {
	private long seq;
	private String name;
	private String addr;
	private Date rdate;
}

@Data
@NoArgsConstructor
@AllArgsConstructor
public class AddressFile {

	private long seqf;
	private String ofname;
	private String sfname;
	private long fsize;
	private long seq;
}
```

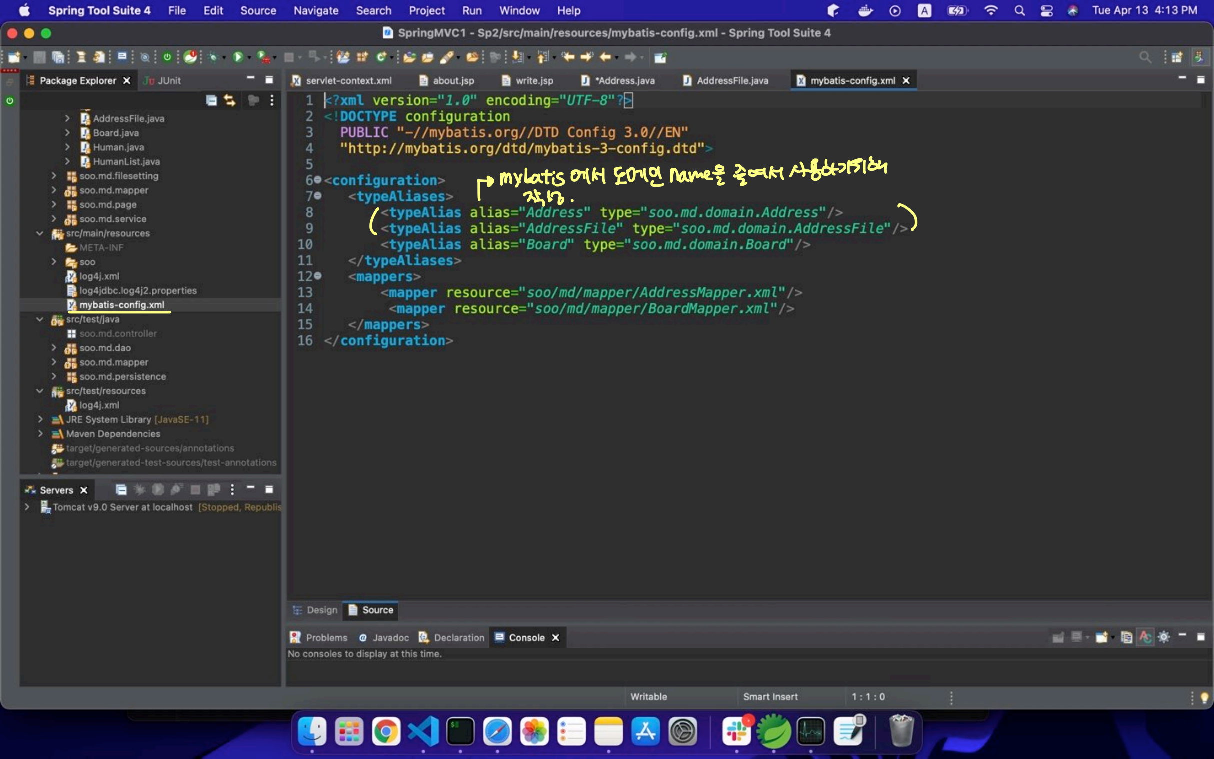
Super table

Sub. table

FK

ALTER TABLE address_file ADD CONSTRAINT ADDRESS_FK FOREIGN KEY(SEQ) REFERENCES address(SEQ) ON DELETE CASCADE;

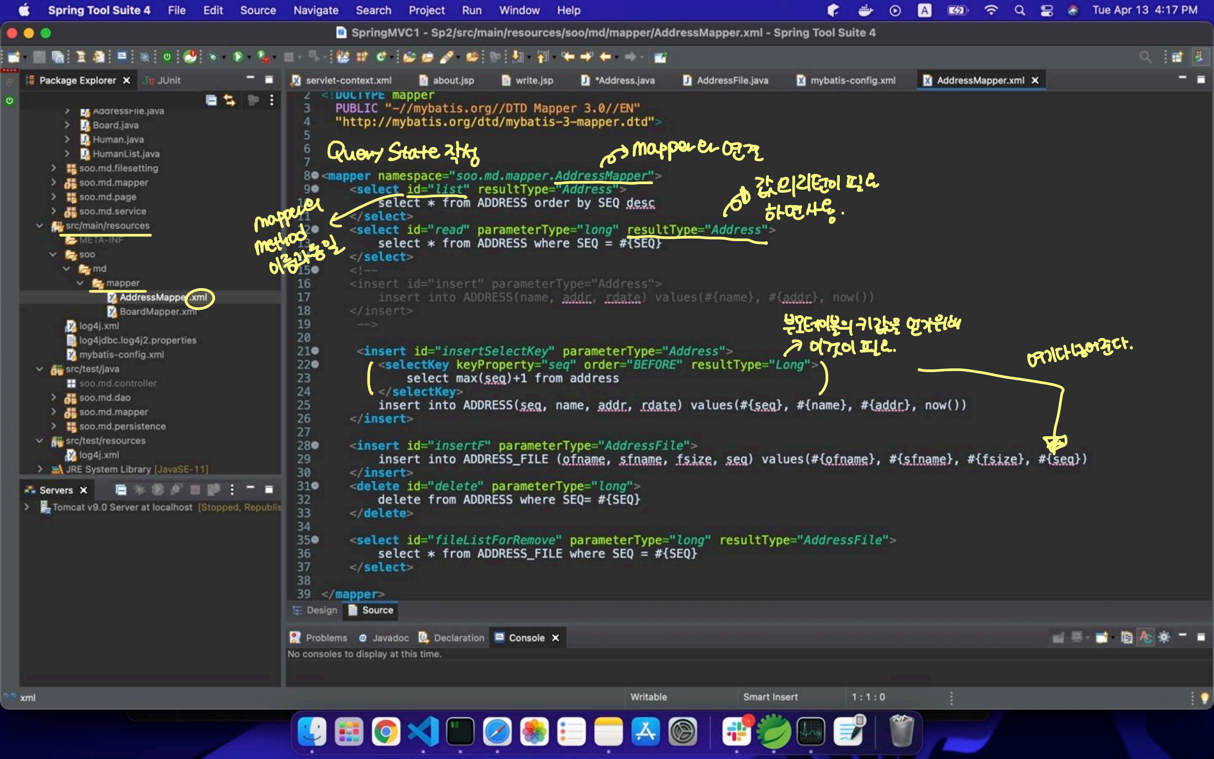
부모테이블 삭제시 자동삭제.



```xml
<?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>
   <typeAliases>
       <typeAlias alias="Address" type="soo.md.domain.Address"/>
       <typeAlias alias="AddressFile" type="soo.md.domain.AddressFile"/>
       <typeAlias alias="Board" type="soo.md.domain.Board"/>
   </typeAliases>
   <mappers>
       <mapper resource="soo/md/mapper/AddressMapper.xml"/>
        <mapper resource="soo/md/mapper/BoardMapper.xml"/>
   </mappers>
</configuration>
```

mybatis 에서 도메인 name을 줄여서 사용하기위해 작성.

No consoles to display at this time.



SpringMVC1 - Sp2/src/main/resources/soo/md/mapper/AddressMapper.xml - Spring Tool Suite 4

Query State 작성

→ mapper와 연결

값의 리턴이 필요하면 사용.

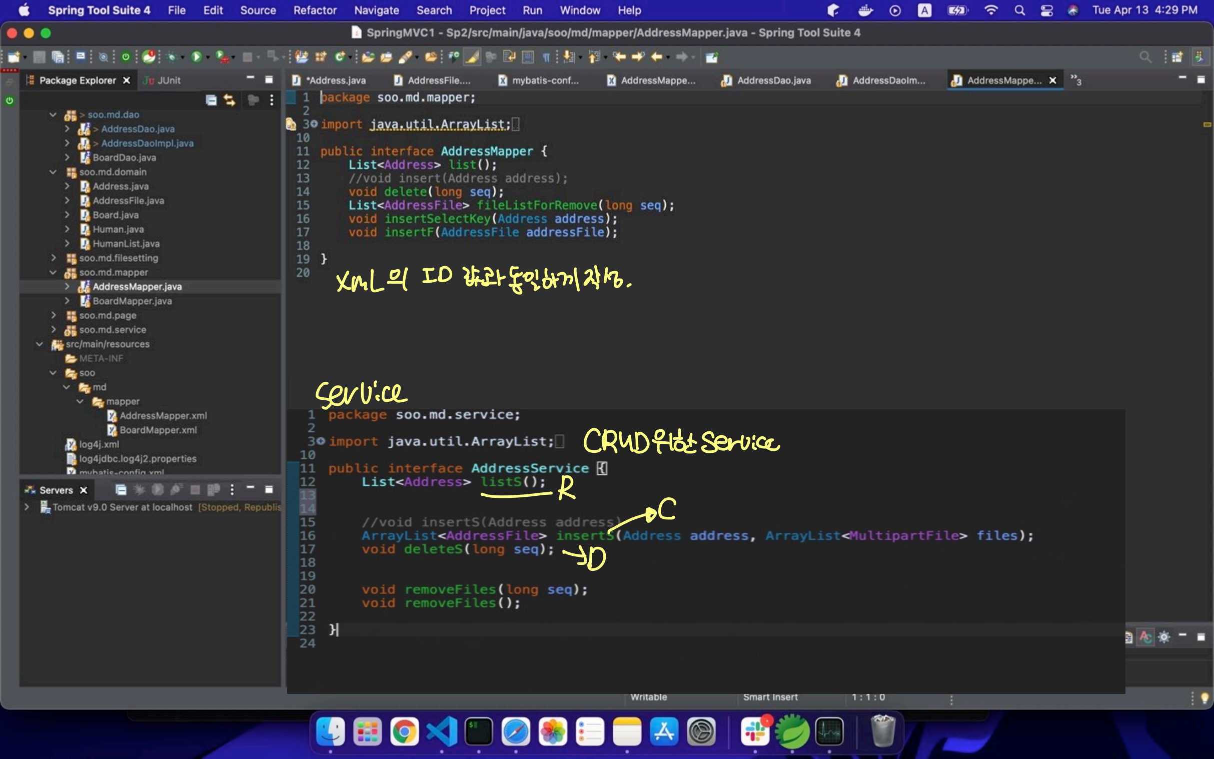
mapper의 method 이름과 동일

부모테이블의 키값을 얻기위해 → 이것이 필요.

여기다 넣어준다.

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



<mapper namespace="soo.md.mapper.AddressMapper">
    <select id="list" resultType="Address">
        select * from ADDRESS order by SEQ desc
    </select>
    <select id="read" parameterType="long" resultType="Address">
        select * from ADDRESS where SEQ = #{SEQ}
    </select>
    <!--
    <insert id="insert" parameterType="Address">
        insert into ADDRESS(name, addr, rdate) values(#{name}, #{addr}, now())
    </insert>
     -->

     <insert id="insertSelectKey" parameterType="Address">
        <selectKey keyProperty="seq" order="BEFORE" resultType="Long">
            select max(seq)+1 from address
        </selectKey>
        insert into ADDRESS(seq, name, addr, rdate) values(#{seq}, #{name}, #{addr}, now())
    </insert>

    <insert id="insertF" parameterType="AddressFile">
        insert into ADDRESS_FILE (ofname, sfname, fsize, seq) values(#{ofname}, #{sfname}, #{fsize}, #{seq})
    </insert>
    <delete id="delete" parameterType="long">
        delete from ADDRESS where SEQ= #{SEQ}
    </delete>

    <select id="fileListForRemove" parameterType="long" resultType="AddressFile">
        select * from ADDRESS_FILE where SEQ = #{SEQ}
    </select>

</mapper>
```

```
No consoles to display at this time.
```



```java
package soo.md.mapper;

import java.util.ArrayList;
public interface AddressMapper {
    List<Address> list();
    //void insert(Address address);
    void delete(long seq);
    List<AddressFile> fileListForRemove(long seq);
    void insertSelectKey(Address address);
    void insertF(AddressFile addressFile);

}
```

XML의 ID 값과 동일하게 작성.

service

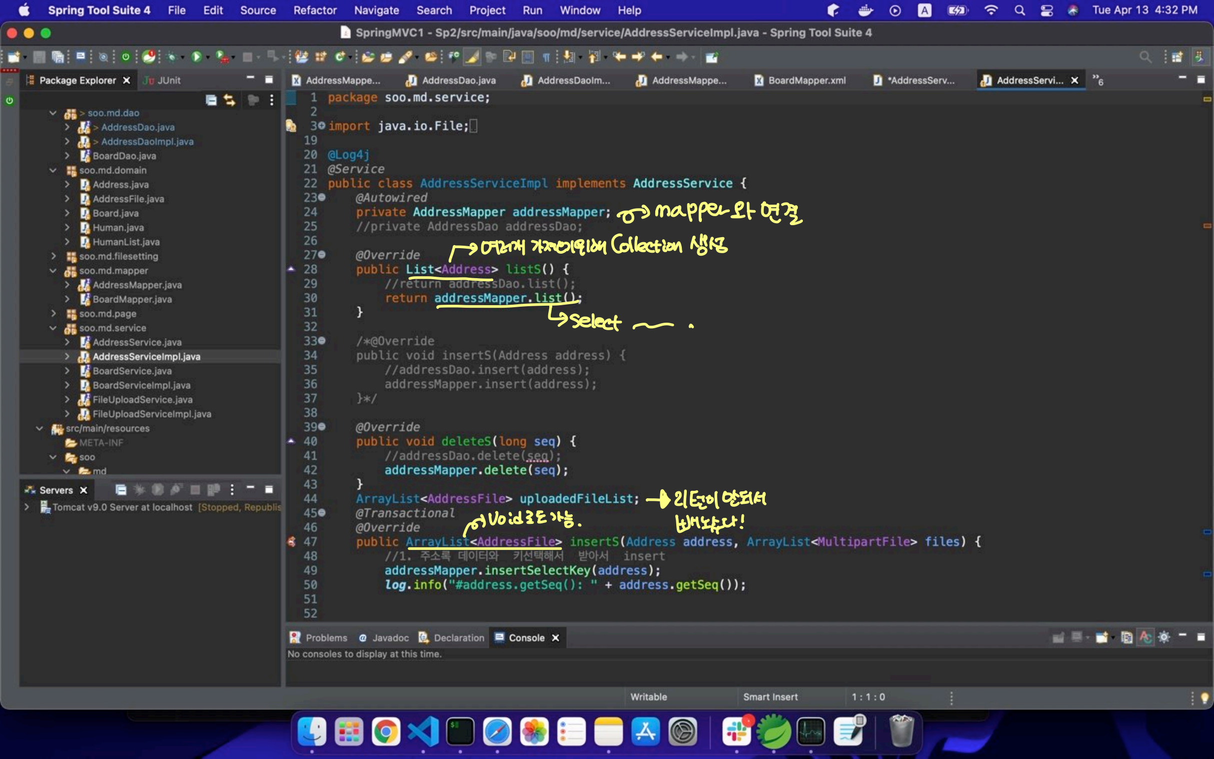
```java
package soo.md.service;

import java.util.ArrayList;
public interface AddressService {
    List<Address> listS();


    //void insertS(Address address)
    ArrayList<AddressFile> insertS(Address address, ArrayList<MultipartFile> files);
    void deleteS(long seq);


    void removeFiles(long seq);
    void removeFiles();

}
```

CRUD 위한 service

R

C

D



```java
package soo.md.service;

import java.io.File;

@Log4j
@Service
public class AddressServiceImpl implements AddressService {
    @Autowired
    private AddressMapper addressMapper;
    //private AddressDao addressDao;

    @Override
    public List<Address> listS() {
        //return addressDao.list();
        return addressMapper.list();
    }

    /*@Override
    public void insertS(Address address) {
        //addressDao.insert(address);
        addressMapper.insert(address);
    }*/

    @Override
    public void deleteS(long seq) {
        //addressDao.delete(seq);
        addressMapper.delete(seq);
    }
    ArrayList<AddressFile> uploadedFileList;
    @Transactional
    @Override
    public ArrayList<AddressFile> insertS(Address address, ArrayList<MultipartFile> files) {
        //1. 주소록 데이터와  키선택해서  받아서  insert
        addressMapper.insertSelectKey(address);
        log.info("#address.getSeq(): " + address.getSeq());


```

No consoles to display at this time.

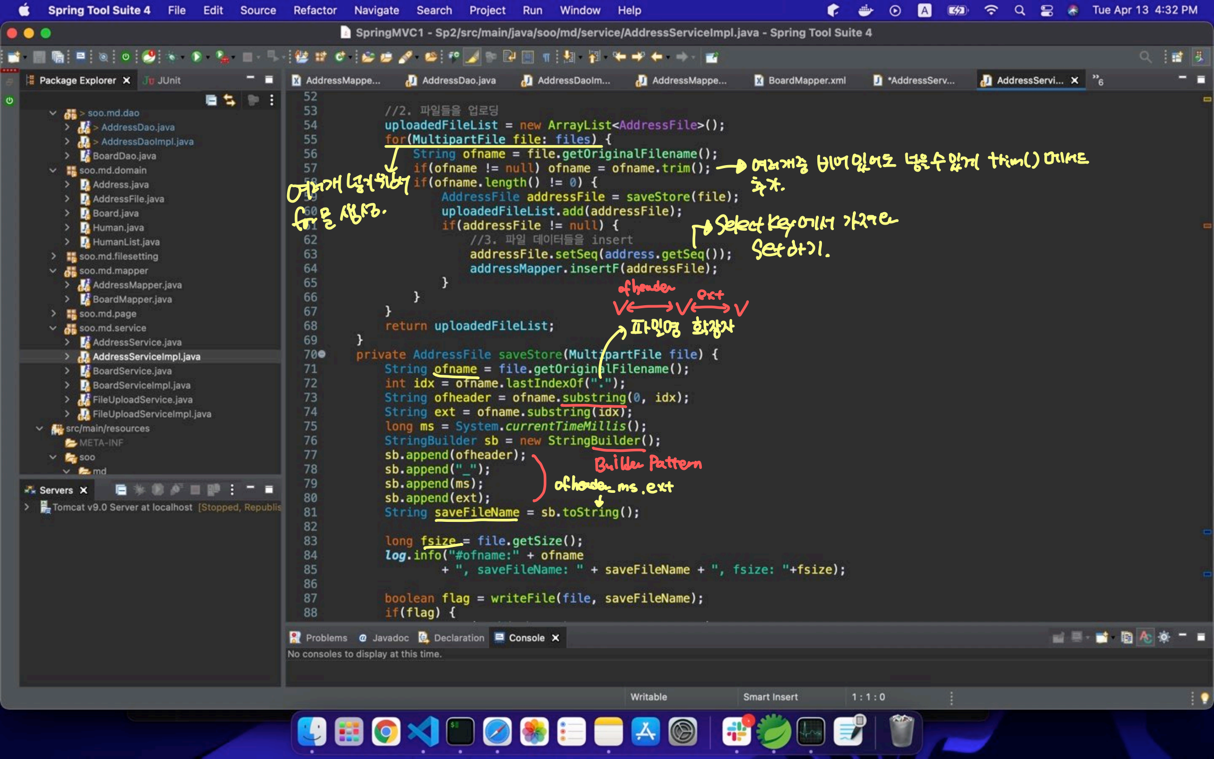
→ mapper와 연결

→ 여러개 가져오기위해 Collection 생성

→ select ~ .

→ 리턴이 안되서 빼놓았다!

→ void로도 가능.



```java

        //2. 파일들을 업로딩
        uploadedFileList = new ArrayList<AddressFile>();
        for(MultipartFile file: files) {
            String ofname = file.getOriginalFilename();
            if(ofname != null) ofname = ofname.trim();
            if(ofname.length() != 0) {
                AddressFile addressFile = saveStore(file);
                uploadedFileList.add(addressFile);
                if(addressFile != null) {
                    //3. 파일 데이터들을 insert
                    addressFile.setSeq(address.getSeq());
                    addressMapper.insertF(addressFile);
                }
            }
        }
        return uploadedFileList;
    }
    private AddressFile saveStore(MultipartFile file) {
        String ofname = file.getOriginalFilename();
        int idx = ofname.lastIndexOf(".");
        String ofheader = ofname.substring(0, idx);
        String ext = ofname.substring(idx);
        long ms = System.currentTimeMillis();
        StringBuilder sb = new StringBuilder();
        sb.append(ofheader);
        sb.append("_");
        sb.append(ms);
        sb.append(ext);
        String saveFileName = sb.toString();

        long fsize = file.getSize();
        log.info("#ofname:" + ofname
                + ", saveFileName: " + saveFileName + ", fsize: "+fsize);

        boolean flag = writeFile(file, saveFileName);
        if(flag) {
```

No consoles to display at this time.

여러개 넣기위해 for문 생성.

여러개중 비어있어도 넣을수있게 trim() 메서드 추가.

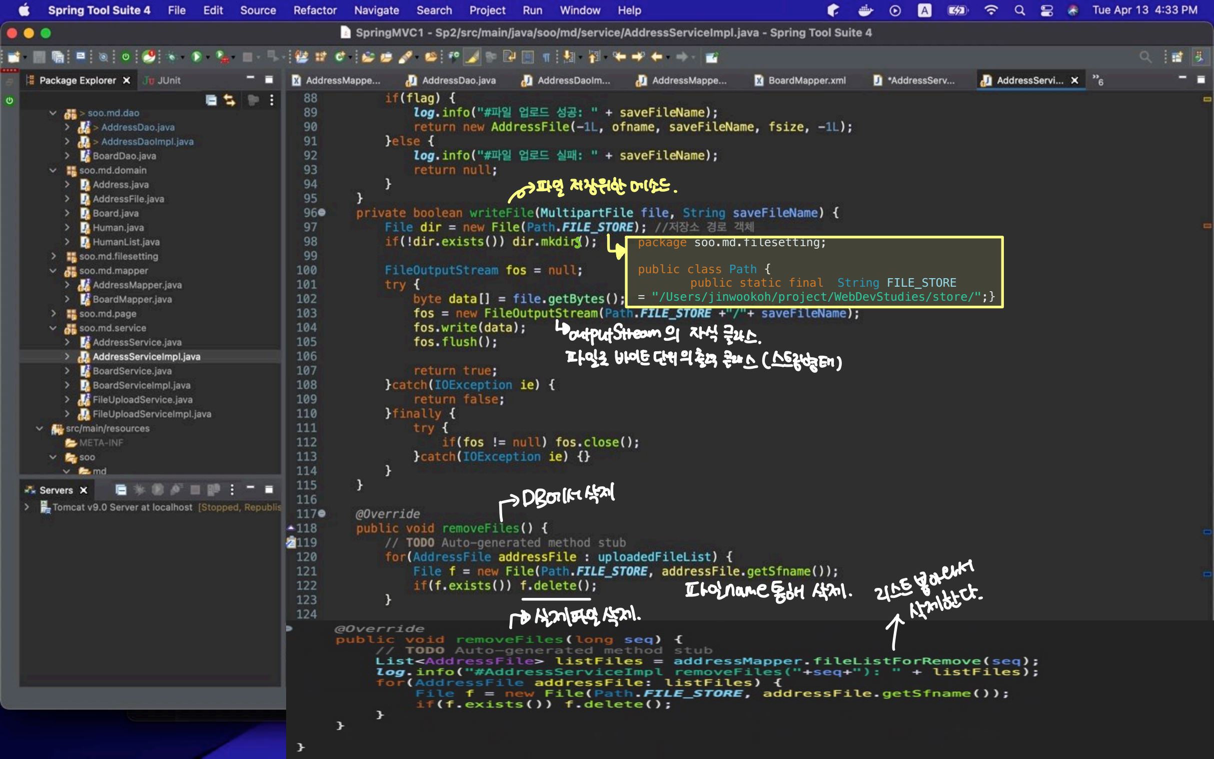
Select Key에서 가져온 Set 하기.

ofheader ext

파일명 확장자

Builder Pattern

ofheader_ms.ext



```java
        if(flag) {
            log.info("#파일 업로드 성공: " + saveFileName);
            return new AddressFile(-1L, ofname, saveFileName, fsize, -1L);
        }else {
            log.info("#파일 업로드 실패: " + saveFileName);
            return null;
        }
    }
    private boolean writeFile(MultipartFile file, String saveFileName) {
        File dir = new File(Path.FILE_STORE); //저장소 경로 객체
        if(!dir.exists()) dir.mkdirs();

        FileOutputStream fos = null;
        try {
            byte data[] = file.getBytes();
            fos = new FileOutputStream(Path.FILE_STORE +"/"+ saveFileName);
            fos.write(data);
            fos.flush();

            return true;
        }catch(IOException ie) {
            return false;
        }finally {
            try {
                if(fos != null) fos.close();
            }catch(IOException ie) {}
        }
    }

    @Override
    public void removeFiles() {
        // TODO Auto-generated method stub
        for(AddressFile addressFile : uploadedFileList) {
            File f = new File(Path.FILE_STORE, addressFile.getSfname());
            if(f.exists()) f.delete();
        }
```

→파일 저장위한 메소드.

```java
package soo.md.filesetting;

public class Path {
        public static final  String FILE_STORE
= "/Users/jinwookoh/project/WebDevStudies/store/";}
```

outputStream의 자식 클래스.
파일로 바이트 단위의 출력 클래스 (스트림형태)

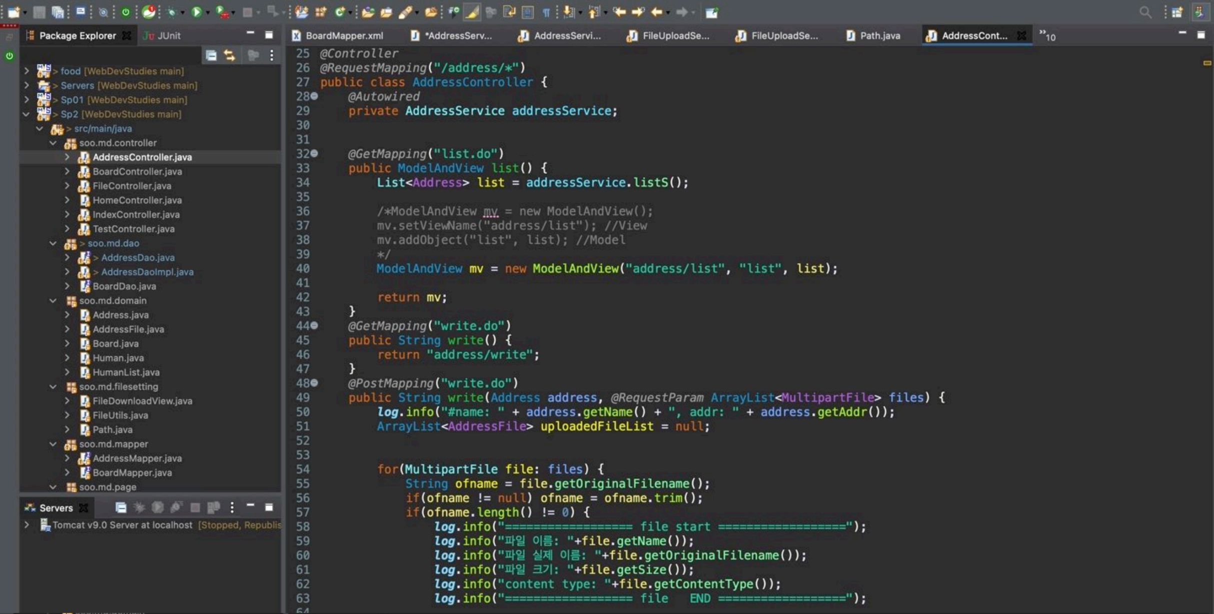
→DB에서 삭제

→삭제파일 삭제.

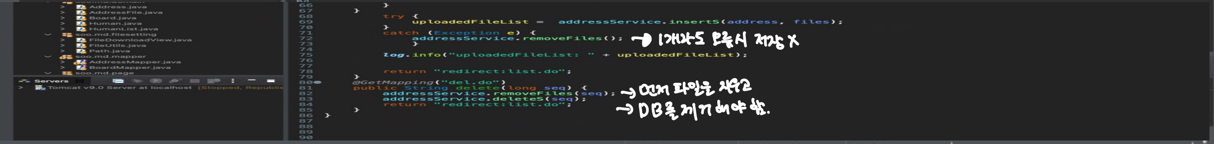
파일name 통해 삭제.

리스트 불러와서 삭제한다.

```java
    @Override
    public void removeFiles(long seq) {
        // TODO Auto-generated method stub
        List<AddressFile> listFiles = addressMapper.fileListForRemove(seq);
        log.info("#AddressServiceImpl removeFiles("+seq+"): " + listFiles);
        for(AddressFile addressFile: listFiles) {
            File f = new File(Path.FILE_STORE, addressFile.getSfname());
            if(f.exists()) f.delete();
        }
    }

}
```



```java
@Controller
@RequestMapping("/address/*")
public class AddressController {
    @Autowired
    private AddressService addressService;


    @GetMapping("list.do")
    public ModelAndView list() {
        List<Address> list = addressService.listS();

        /*ModelAndView mv = new ModelAndView();
        mv.setViewName("address/list"); //View
        mv.addObject("list", list); //Model
        */
        ModelAndView mv = new ModelAndView("address/list", "list", list);

        return mv;
    }
    @GetMapping("write.do")
    public String write() {
        return "address/write";
    }
    @PostMapping("write.do")
    public String write(Address address, @RequestParam ArrayList<MultipartFile> files) {
        log.info("#name: " + address.getName() + ", addr: " + address.getAddr());
        ArrayList<AddressFile> uploadedFileList = null;


        for(MultipartFile file: files) {
            String ofname = file.getOriginalFilename();
            if(ofname != null) ofname = ofname.trim();
            if(ofname.length() != 0) {
                log.info("================== file start ==================");
                log.info("파일 이름: "+file.getName());
                log.info("파일 실제 이름: "+file.getOriginalFilename());
                log.info("파일 크기: "+file.getSize());
                log.info("content type: "+file.getContentType());
                log.info("================== file   END ==================");
```



```java
            }
        }
        try {
            uploadedFileList =  addressService.insertS(address, files);
        }
        catch (Exception e) {
            addressService.removeFiles(); 
            }

        log.info("uploadedFileList: " + uploadedFileList);

        return "redirect:list.do";
    }
    @GetMapping("del.do")
    public String delete(long seq) {
        addressService.removeFiles(seq);
        addressService.deleteS(seq);
        return "redirect:list.do";
    }
}
```

→ 1개라도 오류시 저장 X

→ 먼저 파일을 지우고
→ DB를 제거 해야 함.