



# **spring 5강 – 자료실**

---

**양 명 속**

**[now4ever7@gmail.com]**



# 목차

---

- 파일 업로드 처리
- 자료실
- 답변형 게시판



# 파일 업로드 처리

---



## 파일 업로드 처리

---

- 파일 업로드가 필요한 경우 html 폼의 enctype 속성을 "multipart/form-data" 로 설정해야 함
- 인코딩 타입이 Multipart 인 경우 파라미터나 업로드한 파일을 구하려면 전송 데이터를 알맞게 처리해 주어야 함
- 스프링은 Multipart 지원 기능을 제공하고 있기 때문에, 이 기능을 이용하면 추가적인 처리없이 Multipart 형식으로 전송된 파라미터와 파일 정보를 쉽게 구할 수 있다

# MultipartResolver 설정

- Multipart 지원 기능을 사용하려면 먼저 **MultipartResolver** 를 스프링 설정 파일에 등록해 주어야 함
- **MultipartResolver** 는 Multipart 형식으로 데이터가 전송된 경우, 해당 데이터를 스프링 MVC에서 사용할 수 있도록 변환해줌
  - 예) @PathVariable 어노테이션을 이용해서 Multipart 로 전송된 파라미터와 파일을 사용할 수 있도록 해줌
- 스프링이 기본으로 제공하는 MultipartResolver는 **CommonsMultipartResolver** 이다
  - CommonsMultipartResolver는 Commons FileUpload API 를 이용해서 Multipart를 처리해줌
- CommonsMultipartResolver를 MultipartResolver 로 사용하려면 빈 이름으로 "multipartResolver"를 사용해서 등록하면 됨

```
<bean id="multipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
</bean>
```

DispatcherServlet은 이름이 "multipartResolver"인 빈을 사용하기 때문에 다른 이름을 지정할 경우 MultipartResolver 로 사용되지 않는다

# MultipartResolver 설정

- CommonsMultipartResolver 클래스의 프로퍼티

[표 6.4] CommonsMultipartResolver 클래스의 프로퍼티

프로퍼티	타입	설명
maxUploadSize	long	최대 업로드 가능한 바이트 크기. -1은 제한이 없음을 의미한다. 기본 값은 -1 이다.
maxInMemorySize	int	디스크에 임시 파일을 생성하기 전에 메모리에 보관할 수 있는 최대 바이트 크기. 기본 값은 10240 바이트이다.
defaultEncoding	String	요청을 파싱할 때 사용할 캐릭터 인코딩. 지정하지 않을 경우, HttpServletRequest.setCharacterEncoding() 메서드로 지정된 캐릭터 셋이 사용된다. 아무 값도 없을 경우 ISO-8859-1을 사용한다.

```
<!-- MultipartResolver -->
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="1024000"></property>
</bean>
```



## @RequestParam 어노테이션을 이용한 업로드 파일 접근

- [1] 업로드한 파일을 전달받는 첫 번째 방법
  - @RequestParam 어노테이션이 적용된 **MultipartFile** 타입의 파라미터를 사용하는 것
  - 예) html 입력폼이 다음과 같이 작성되어 있다면.

```
<form action="submitReport1.do" method="post" enctype="multipart/form-data">  
    학번: <input type="text" name="studentNumber" />  
    <br/>  
    리포트파일: <input type="file" name="report" />  
    <br/>  
    <input type="submit" />  
</form>
```

- 파일은 report 파라미터를 통해서 전달됨
- 이 경우 @RequestParam 어노테이션과 MultipartFile 타입의 파라미터를 이용해서 업로드 파일 데이터를 전달받을 수 있다

# @RequestParam 어노테이션을 이용한 업로드 파일 접근

```
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

@Controller
public class ReportSubmissionController {

    @RequestMapping(value = "/report/submitReport1.do", method = RequestMethod.POST)
    public String submitReport1(
        @RequestParam("studentNumber") String studentNumber,
        @RequestParam("report") MultipartFile report) {

        //MultipartFile이 제공하는 메서드를 이용해서 업로드 데이터 접근
        printInfo(studentNumber, report);
        return "report/submissionComplete";
    }
}
```

- **MultipartFile** 인터페이스는 스프링에서 업로드 한 파일을 표현할 때 사용되는 인터페이스로서, MultipartFile 인터페이스를 이용해서 업로드한 파일의 이름, 실제 데이터, 파일 크기 등을 구할 수 있다





# MultipartHttpServletRequest를 이용한 업로드 파일 접근

- [2] 업로드한 파일을 전달받는 두 번째 방법
  - **MultipartHttpServletRequest** 인터페이스를 사용하는 것

```
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.multipart.MultipartHttpServletRequest;

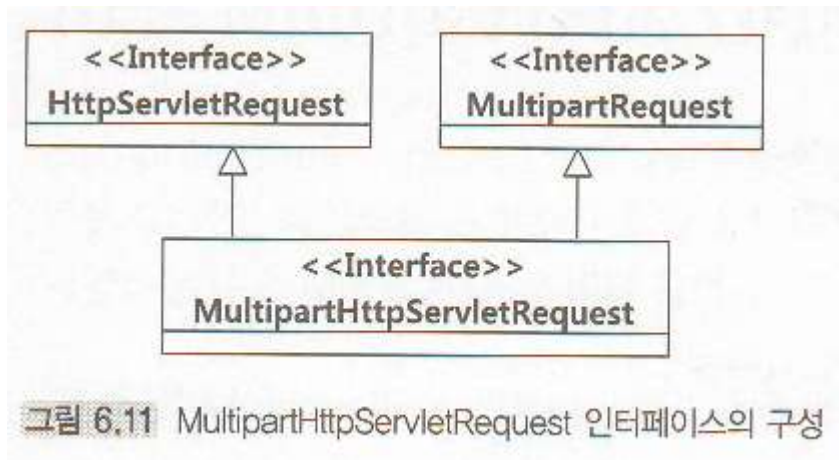
@Controller
public class ReportSubmissionController {

    @RequestMapping(value = "/report/submitReport2.do", method = RequestMethod.POST)
    public String submitReport2(MultipartHttpServletRequest request) {
        String studentNumber = request.getParameter("studentNumber");
        MultipartFile report = request.getFile("report");
        printInfo(studentNumber, report);
        return "report/submissionComplete";
    }
}
```

- MultipartHttpServletRequest 인터페이스는 스프링이 제공하는 인터페이스로서, **Multipart 요청이 들어올 때 내부적으로 원본 HttpServletRequest 대신 사용되는 인터페이스임**

# MultipartHttpServletRequest를 이용한 업로드 파일 접근

- MultipartHttpServletRequest 인터페이스는 실제로는 어떤 메서드도 선언하고 있지 않으며, HttpServletRequest 인터페이스와 MultipartRequest 인터페이스를 상속받고 있다



- MultipartHttpServletRequest 인터페이스는 javax.servlet.HttpServletRequest 인터페이스를 상속받기 때문에 웹 요청 정보를 구하기 위한 getParameter()나 getHeader()와 같은 메서드를 사용할 수 있으며, 추가로 MultipartRequest 인터페이스가 제공하는 Multipart 관련 메서드를 사용할 수 있음

# MultipartHttpServletRequest를 이용한 업로드 파일 접근

- MultipartHttpServletRequest 인터페이스의 파일 관련 주요 메서드

[표 6.5] MultipartRequest 인터페이스의 파일 관련 주요 메서드

메서드	설 명
Iterator<String> getFileNames( )	업로드 된 파일들의 이름 목록을 제공하는 Iterator를 구한다.
MultipartFile <u>getFile</u> (String name)	파라미터 이름이 name인 업로드 파일 정보를 구한다.
List<MultipartFile> getFiles(String name)	파라미터 이름이 name인 업로드 파일 정보 목록을 구한다.
Map<String, MultipartFile> <u>getFileMap</u> ( )	파라미터 이름을 키로 파라미터에 해당하는 파일 정보를 값으로 하는 Map을 구한다.

# 커맨드 객체를 통한 업로드 파일 접근

- [3] 커맨드 객체를 이용해도 업로드 한 파일을 전달받을 수 있음
  - 단지 커맨드 클래스에 파라미터와 동일한 이름의 MultipartFile 타입 프로퍼티를 추가해주기만 하면 됨
  - 예) 업로드 파일의 파라미터 이름이 "report"인 경우, "report" 프로퍼티를 커맨드 클래스에 추가해주면 됨

```
import org.springframework.web.multipart.MultipartFile;
public class ReportCommand {
    private String studentNumber;
    private MultipartFile report;

    public String getStudentNumber() {
        return studentNumber;
    }
    public void setStudentNumber(String studentNumber) {
        this.studentNumber = studentNumber;
    }
    public MultipartFile getReport() {
        return report;
    }
    public void setReport(MultipartFile report) {
        this.report = report;
    }
}
```

MultipartFile 타입의 프로퍼티를 커맨드 클래스에 추가해주었으면, @RequestMapping 메서드의 커맨드 객체로 사용함으로써 업로드 파일 정보를 커맨드 객체를 통해서 전달받을 수 있게 됨



# 커맨드 객체를 통한 업로드 파일 접근

```
@Controller
public class ReportSubmissionController {

    @RequestMapping(value = "/report/submitReport3.do", method = RequestMethod.POST)
    public String submitReport3(ReportCommand reportCommand) {
        printInfo(reportCommand.getStudentNumber(), reportCommand.getReport());
        return "report/submissionComplete";
    }
}
```

# MultipartFile 인터페이스 사용

- org.springframework.web.multipart.MultipartFile 인터페이스는 업로드 한 파일 정보 및 파일 데이터를 표현하기 위한 용도로 사용됨
- MultipartFile 인터페이스가 제공하는 주요 메서드

[표 6.6] MultipartFile 인터페이스의 주요 메서드

메서드	설 명
String getName( )	파라미터 이름을 구한다.
String getOriginalFilename( )	업로드 한 파일의 이름을 구한다.
boolean <u>isEmpty( )</u>	업로드 한 파일이 존재하지 않는 경우 true를 리턴한다.
long getSize( )	업로드 한 파일의 크기를 구한다.
byte[ ] getBytes( ) throws IOException	업로드 한 파일 데이터를 구한다.
InputStream getInputStream( ) throws IOException	업로드 한 파일 데이터를 읽어오는 InputStream을 구한다. InputStream의 사용이 끝나면 알맞게 종료해 주어야 한다.
void transferTo(File dest) throws IOException	업로드 한 파일 데이터를 지정한 파일에 저장한다.



# MultipartFile 인터페이스 사용

- 업로드한 파일 데이터를 구하는 가장 단순한 방법은 **MultipartFile.getBytes()** 메서드를 이용하는 것
  - 바이트 배열을 구한 뒤에 파일이나 DB 등에 저장하면 됨

```
if(!multipartFile.isEmpty()){  
    byte[] fileData = multipartFile.getBytes();  
    //byte 배열을 파일/DB/네트워크 등으로 전송  
    ...  
}
```

- 업로드한 파일 데이터를 특정 파일로 저장하고 싶다면 **MultipartFile.transferTo()** 메서드를 사용하는 것이 편리함

```
if(!multipartFile.isEmpty()){  
    File file=new File(fileName);  
    multipartFile.transferTo(file);  
    ...  
}
```



# 자료실

---





# 파일 업로드를 위한 폼 형태

- 웹 브라우저를 통해서 파일을 전송하기 위한 폼 구성
  - [1] 파일을 전송하기 위한 파일 선택창을 사용하기 위해 `<input type="file">` 태그 사용
  - [2] 선택된 파일을 업로드하기 위해서는 form 태그의 `method="post"`, `enctype="multipart/form-data"` 지정

```
<form name="frm" method="post" enctype="multipart/form-data">  
    <input type="file" name="filename">  
</form>
```

- post방식일 경우 2가지 인코딩
  - [1] `application/x-www-form-urlencoded` : 디폴트, 파일 이름만 전송됨
  - [2] `multipart/form-data` : 파일 이름과 함께 파일 데이터가 전송됨

- 글 쓰기 페이지 -write.jsp  
글 쓰는 폼에 업로드할 파일을 선택할 수 있도록 [file] 컨트롤 추가

## 제 목

작성자

비밀번호

이메일

## 내용

## 첨부파일

 (최대 2M)

U/I

## 글목록



# write.jsp

---

```
<form name="frm1" method="post" action="<c:url value='/reboard/write.do'/>"
      onsubmit="return send(this)" enctype="multipart/form-data">
  <fieldset>
    <legend>글쓰기</legend>
    .....
    <p>
      <label for="content">내용</label>
      <textarea name="content" id="content" rows="12" cols="60" ></textarea>
    </p>
    <p>
      <label for="uploadFile">첨부파일</label>
      <input type="file" name="uploadFile" id="uploadFile" size="36">(최대 2M)
    </p>
    <p class="center">
      <input type="submit" value="등록">
      <input type="Button" value="글목록"
        onclick="location.href='<c:url value='/reboard/list.do'/>'" >
    </p>
  </fieldset>
</form>
```



# ReboardVo

---

//답변형 게시판 추가

```
private int groupNo;  
private int step;  
private int sortNo;  
private String delFlag;
```

//자료실 추가

```
private String fileName;  
private long fileSize;  
private int downCount;
```



# fileUpload.properties

---

file.upload.path=pds\_upload

file.upload.path.test=d:\\Ws\\springherb\\src\\main\\webapp\\pds\_upload

imageFile.upload.path=pd\_images

imageFile.upload.path.test=d:\\Ws\\springherb\\src\\main\\webapp\\pd\_images

file.upload.type=test

#file.upload.type=deploy



# <util:properties >

- 스프링은 List, Set, Map, Properties 와 같은 컬렉션 타입을 XML 로 작성해서 프로퍼티에 주입하는 방법을 제공함
- [1] List, Set
  - <list>와 <value>를 이용해 선언함
  - 프로퍼티 타입이 다음과 같다면
    - List<String> names;
  - 다음과 같이 names 프로퍼티 값을 리스트로 선언할 수 있다

```
<property name="names">
  <list>
    <value>Spring</value>
    <value>JSP</value>
  </list>
</property>
```

- 프로퍼티가 Set 이라면 <list> 대신 <set>을 사용하면 됨



# <util:properties >

## ■ [2] Map

- 맵은 <map>과 <entry> 이용
  - Map<String, Integer> ages;

```
<property name="ages">
  <map>
    <entry key="kim" value="20"/>
    <entry key="lee" value="25"/>
  </map>
</property>
```

XML에 위와 같이 선언해주면 **Map** 타입의 오브젝트로 변환되어 프로퍼티에 주입된다

## ■ [3] Properties

- java.util.Properties 타입은 <props>, <prop>를 이용
  - Properties settings;

```
<property name="settings">
  <props>
    <prop key="username">hong</prop>
    <prop key="pwd">1234</prop>
  </props>
</property>
```

<util:list >  
<util:set >  
<util:map >

# <util:properties >

- 컬렉션을 프로퍼티의 값으로 선언하는 대신 독립적인 빈으로 만들 수도 있다
- 이때는 컬렉션 오브젝트가 아이디를 가진 빈이 되므로 여러 빈에서 공통적으로 참조할 수도 있다
- 컬렉션을 별도로 선언할 때는 util 스키마의 전용 태그를 활용하면 됨
- <util:properties >
  - Properties 는 <util:properties >를 이용

```
<util:properties id="settings">  
    <prop key="username">hong</prop>  
    <prop key="pwd">1234</prop>  
</util:properties>
```

Properties 는 XML에 직접 내용을 등록하는 대신 외부의 프로퍼티 파일을 지정해서 그 내용을 사용할 수 있다.

```
<util:properties id="fileUploadProperties"  
    location="classpath:/config/props/fileUpload.properties" />
```





## <util:properties >

- <util:properties > 태그를 이용하면 프로퍼티 파일을 읽어서 Properties 타입의 빈으로 만들 수 있다
- 다음과 같이 선언하면 fileUpload.properties 파일의 내용을 읽어서 Properties 안에 담아 빈으로 생성해준다

```
<util:properties id="fileUploadProperties"  
                location="classpath:/config/props/fileUpload.properties" />
```

- <util:properties >는 <context:property-placeholder> 처럼 빈 팩토리 후처리로 동작해서 빈의 값을 변경하는 기능을 가진 것은 아니다
- 단순히 프로퍼티 파일의 내용을 담은 Properties 타입 빈을 만들어줄 뿐이다.

# PropertyPlaceholderConfigurer

- 프로퍼티 파일을 이용한 값 설정

- 수동 변환 :PropertyPlaceholderConfigurer

- 프로퍼티 치환자(placeholder)를 이용하는 방법
    - 프로퍼티 치환자는 프로퍼티 파일의 키 값을 \${ } 안에 넣어서 만들어 준다

```
<context:property-placeholder location="classpath:/config/props/database.properties" />

<bean id="dataSource-mysql" class="org.apache.commons.dbcp.BasicDataSource">
    <property name="driverClassName" value="${Globals.DriverClassName}"/>
</bean>
```

- 스프링 컨테이너는 초기화 작업 중에 database.properties 파일을 읽고, 각 키에 \${} 를 붙인 값과 동일한 value 선언을 찾는다.
    - 그리고 발견된 value 의 값을 프로퍼티 파일에 정의해둔 값으로 바꿔치기한다.
  - 프로퍼티 치환자를 이용해 설정 값을 프로퍼티 파일로 분리해두면 DB 연결 정보가 변경되더라도 XML을 수정하는 대신 database.properties 파일만 수정해주면 된다
  - 또는 개발환경, 테스트환경, 운영환경에 각각 다른 database.properties 파일을 두면 환경에 따라 다른 DB 연결정보를 이용할 수 있다.



# context-common.xml

---

```
<bean id="spring.RegularCommonsMultipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="100000000" />
    <property name="maxInMemorySize" value="100000000" />
</bean>
```

```
<alias name="spring.RegularCommonsMultipartResolver" alias="multipartResolver" />
```

```
<util:properties id="fileUploadProperties"
    location="classpath:/config/props/fileUpload.properties" />
```

프로퍼티 파일(fileUpload.properties)을 읽어서 **Properties** 타입의 빈으로 생성해준다.

```
@Resource(name = "fileUploadProperties")
Properties fileuploadProperties;
```



# servlet-context.xml

---

- 첨부 파일 다운로드 처리를 위해 변경 및 추가한다

```
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
    <beans:property name="prefix" value="/WEB-INF/views/" />  
    <beans:property name="suffix" value=".jsp" />  
    <beans:property name="order" value="1" />  
</beans:bean>
```

```
<beans:bean class="org.springframework.web.servlet.view.BeanNameViewResolver">  
    <beans:property name="order" value="0" />  
</beans:bean>
```

```
<beans:bean id="downloadView"  
    class="com.herb.app.reboard.controller.ReBoardDownloadView" />
```

```

public class Utility {
    public static String getUniqueFileName(String dirPath, String fileName){
        //test.txt => test_1.txt => test_2.txt
        //[1] 순수 파일명만 가져오기 test
        String fName = fileName.substring(0, fileName.lastIndexOf(".")); //test
        //[2] 순수 확장자만 가져오기(.포함) .txt
        String ext=fileName.substring(fileName.lastIndexOf(".")); //.확장자만
        System.out.println("fName:"+fName+", ext:"+ext );

        //같은 이름의 파일 존재여부, 우선 있다고 가정.
        boolean bExist = true;
        int count = 0;
        while (bExist) {
            //D:WWWsiteWWWuploadedWWWtest.txt
            if (new File(dirPath, fileName).exists()){
                count++;
                fileName = fName + "_" + count + ext; //test_1.txt
                System.out.println("fName:"+fName+", ext:"+ext +", newFileName :"+ fileName);
            }else{
                bExist = false;
            }
        }

        return fileName;
    }
}

```

test.txt => **test\_1.txt** => **test\_2.txt**

```

/**
 * 시스템에서 17자리의TIMESTAMP값을 구한다
 * @return
 */
private static String getTimeStamp() {
    String result = "";

    // 문자열로 변환하기 위한 패턴(년-월-일 시:분:초:밀리초(자정 이후 초))
    String pattern = "yyyyMMddhhmmssSSS";

    //SimpleDateFormat sdf = new SimpleDateFormat(pattern, Locale.KOREA);
    SimpleDateFormat sdf = new SimpleDateFormat(pattern);
    /*Timestamp ts = new Timestamp(System.currentTimeMillis());
    result = sdf.format(ts.getTime());*/

    Date today = new Date();
    result = sdf.format(today);
    //result = sdf.format(today.getTime());
    System.out.println("getTimeStamp():"+result);

    return result;
}

```

```
public String getUniqueFileName(String fileName){
    //파일명이 중복될 경우 파일이름 변경하기
    //파일명에 현재시간을 붙여서 변경된 파일이름 구하기
    //a.txt => a_20150519123315235.txt

    //순수파일명만 구하기
    int idx = fileName.lastIndexOf(".");
    String fileNm=fileName.substring(0, idx); //a

    //확장자 구하기
    String ext = fileName.substring(idx); //.txt

    //변경된 파일 이름
    String result = fileNm+"_"+ getTimestamp() +ext;
    logger.info("변경된 파일이름:{}, result)", result);

    return result;
}
```

@Service  
@Transactional

```
public class ReBoardServiceImpl implements ReboardService{
```

```
    @Autowired ReboardDAO reBoardDao;
```

```
    //자료실 업로드 폴더
```

```
    @Resource(name = "fileUploadProperties")
```

```
    Properties fileuploadProperties;
```

```
<util:properties id="fileUploadProperties"
    location="classpath:/config/props/fileUpload.properties" />
```

프로퍼티 파일(fileUpload.properties)을 읽어서  
Properties 타입의 빈으로 생성해준다.

```
private static final Logger logger = LoggerFactory.getLogger(ReBoardServiceImpl.class);
```

```
    //파일 업로드 처리
```

```
public List<Map<String, Object>> fileupload(HttpServletRequest request) throws Exception{
```

```
    final MultipartHttpServletRequest multiRequest = (MultipartHttpServletRequest) request;
```

```
    // extract files
```

```
    final Map<String, MultipartFile> filesMap = multiRequest.getFileMap();
```

```
    //final List<MultipartFile> files = multiRequest.getFiles(null);
```

```
    // process files
```

```
    String uploadLastPath = fileuploadProperties.getProperty("file.upload.path");
```

```
    String uploadPath1 = request.getSession().getServletContext().getRealPath(uploadLastPath);
```

```
    String uploadPath = fileuploadProperties.getProperty("file.upload.path.test");
```

```
    logger.debug("uploadPath1 ={}", uploadPath={}, uploadPath1, uploadPath);
```



```
File saveFolder = new File(uploadPath);
String fileName = null;
```

```
// 디렉토리 생성
```

```
if (!saveFolder.exists() || saveFolder.isFile()) {
    saveFolder.mkdirs();
}
```

```
/*Iterator<String> keyIter = filesMap.keySet().iterator();
while (keyIter.hasNext())
{
```

```
    String key = keyIter.next();
    MultipartFile tmpFile = filesMap.get(key);
```

```
}//while
*/
```

```
List<Map<String, Object>> resultList = new ArrayList<Map<String, Object>>();
```

```
Iterator<Entry<String, MultipartFile>> itr = filesMap.entrySet().iterator();
```

```
MultipartFile tempFile;
```

```
while (itr.hasNext()) {
```

```
    Entry<String, MultipartFile> entry = itr.next();
```

```
    tempFile = entry.getValue();
```

```
    if(!tempFile.isEmpty()){
```

```
        long fileSize = tempFile.getSize(); //파일 크기
```

```
        String oName = tempFile.getOriginalFilename();
```

업로드된 파일을 임시파일 형태로 제공

```

        //변경된 파일 이름
        fileName = Utility.getUniqueFileName(uploadPath, oName);
        logger.debug("fileSize ={}", fileName={}, fileSize, fileName);

        // 파일 전송
        File myfile = new File(uploadPath, fileName);
        tempFile.transferTo(myfile);

        Map<String, Object> resultMap = new HashMap<String, Object>();
        resultMap.put("fileName",fileName);
        resultMap.put("fileSize",fileSize);

        resultList.add(resultMap);

    }//if
} //while

return resultList;
}

```

ReBoardServiceImpl

```
public String getUploadPath(HttpServletRequest request){
    String uploadPath="";

    String upType= fileuploadProperties.getProperty("file.upload.type");

    if(upType.equals("test")){
        uploadPath= fileuploadProperties.getProperty("file.upload.path.test");
    }else{
        String upPath= fileuploadProperties.getProperty("file.upload.path");
        uploadPath=request.getSession().getServletContext().getRealPath(upPath);
    }

    logger.debug("upType={}", uploadPath={}, upType, uploadPath);

    return uploadPath;
}
```



# @Resource

- XML 대신 애노테이션을 이용해 빈의 의존관계를 정의할 수 있는 두 가지 방법
  - [1] @Autowired
    - 기본적으로 타입에 의한 자동 와이어링 방식으로 동작함
    - XML의 타입에 의한 자동와이어링 방식을 생성자, 필드, 수정자 메서드 (setter), 일반 메서드의 4가지로 확장한 것
    - @Resource 와 다른점은 이름 대신 필드나 프로퍼티 타입을 이용해 후보빈을 찾는다는 것

```
public class Hello{  
    @Autowired  
    private Printer printer;
```

필드의 타입이 **Printer** 이므로 현재 등록된 빈에서 **Printer** 타입에 대입 가능한 빈을 찾는다

이름을 이용해 빈을 지정하고 싶다면 **@Resource** 를 사용하고, 타입과 한정자를 활용하고 싶을 때는 **@Autowired** 를 사용하는 것이 바람직하다

# @Resource

```
@Repository("daoMybatis")
public class ReBoardDAOMybatis implements ReboardDAO {
-----
//@Autowired private ReboardDAO reBoardDao;
//@Resource(name="reBoardDAOMybatis")
@Resource(name="daoMybatis")
private ReboardDAO reBoardDao;
```

## ■ [2] @Resource

- 기본적으로 참조할 **빈의 이름을 이용**해서 빈을 찾는다
- <property> 선언과 비슷하게 주입할 빈을 아이디로 지정하는 방법
- @Resource 는 자바 클래스의 수정자(setter) 뿐만 아니라 필드에도 붙일 수 있다

```
public class Hello{
    private Printer printer;
    ...
    @Resource(name="printer") // <property name="printer" ref="printer" /> 와 동일
    public void setPrinter(Printer printer){
        this.printer=printer;
    }
}
```

```
public class Hello{
    @Resource(name="printer")
    private Printer printer;
```

```
public class Hello{
    @Resource
    private Printer printer;
```

참조하는 빈의 이름을 생략할 수 도 있다  
**name** 엘리먼트를 생략하면 **DI**할 빈의 이름이 프로퍼티나 필드 이름과 같다고 가정함

```
@Controller
@RequestMapping("/reboard")
public class ReboardController {
    @Autowired ReboardService reBoardService;

    @Resource(name = "fileUploadProperties")
    Properties fileuploadProperties;

    @Resource(name="pagingProperties")
    Properties pagingProperties;

    private static final Logger logger = LoggerFactory.getLogger(ReboardController.class);

    @RequestMapping(value="/write.do", method=RequestMethod.GET)
    public String insertGet(){
        //글 등록 화면 보여주기
        return "/reboard/write";
    }
    @RequestMapping(value="/write.do", method=RequestMethod.POST)
    public String insertPost(ReBoardBean reBoardBean, HttpServletRequest request){
        //[등록]버튼 클릭시 submit된 경우 글쓰기 처리 - insert

        //1. 파라미터
        logger.debug("파라미터 reBoardBean ={}", reBoardBean);

        //파일 업로드 처리
        String fileName="";
        long fileSize=0;
```

```

List<Map<String, Object>> resultList = null;
try {
    resultList = reBoardService.fileupload(request);
    for(int i=0;i<resultList.size();i++){
        Map<String, Object> fileInfoMap = resultList.get(i);

        fileName = (String) fileInfoMap.get("fileName");
        fileSize =(Long) fileInfoMap.get("fileSize");
    }//for

    logger.debug("파일 업로드 성공, fileName={}, fileSize={}", fileName, fileSize);
} catch (Exception e) {
    logger.debug("파일 업로드 실패");
    e.printStackTrace();
}
reBoardBean.setFileName(fileName);
reBoardBean.setFileSize(fileSize);

//2. db작업
int n = reBoardService.insertReBoard(reBoardBean);
logger.debug("글 등록 결과 : n=" + n);

//3. 결과, 뷰페이지 저장
return "redirect:/reboard/list.do";
}

```

```
@RequestMapping(value="/reply.do", method=RequestMethod.GET)
public String replyGet(@RequestParam("no") int no, Model model){
    //답변달기 화면 보여주기
    //1. 파라미터
    logger.debug("파라미터, no={}", no);

    //2. db작업-select
    ReBoardBean bean=reBoardService.selectByNo(no);

    //3. 결과 저장, 리턴
    model.addAttribute("bean", bean);
    return "/reboard/reply";
}
```

```
@RequestMapping(value="/reply.do", method=RequestMethod.POST)
public String replyPost(ReBoardBean bean){
    //답변달기 처리
    //1. 파라미터
    logger.debug("파라미터, bean={}", bean);

    //2. db작업 - insert
    int n= reBoardService.replyReBoard(bean);
    logger.debug("답변달기 결과, n={}", n);

    //3. 결과 저장, 리턴
    return "redirect:list.do";
}
```



@RequestMapping("/list.do")

public String listReBoard(HttpServletRequest request, SearchBean searchBean, Model model){

//전체 글 목록 조회, 검색 - select

//1. 파라미터 - 검색관련

logger.debug("파라미터 bean:{}", searchBean);

recordCountPerPage=5  
blockSize=10

//2. db작업 - select

int recordCountPerPage = Integer.parseInt(pagingProperties.getProperty("recordCountPerPage"));

int blockSize = Integer.parseInt(pagingProperties.getProperty("blockSize"));

logger.debug("recordCountPerPage={}, blockSize={}", recordCountPerPage, blockSize);

searchBean.setBlockSize(blockSize);

searchBean.setRecordCountPerPage(recordCountPerPage);

/\*\* paging setting \*/

PaginationInfo paginationInfo = new PaginationInfo();

paginationInfo.setCurrentPageNo(searchBean.getCurrentPage());

paginationInfo.setRecordCountPerPage(searchBean.getRecordCountPerPage());

paginationInfo.setBlockSize(searchBean.getBlockSize());

searchBean.setFirstRecordIndex(paginationInfo.getFirstRecordIndex());

searchBean.setLastRecordIndex(paginationInfo.getLastRecordIndex());

List<ReBoardBean> reboardList = reBoardService.selectAll(searchBean);

logger.debug("전체 조회 결과 reboardList.size()={}", reboardList.size());

int totalRecord = reBoardService.selectTotalRecord(searchBean);

paginationInfo.setTotalRecordCount(totalRecord);

//3. 결과, 뷰페이지 저장

model.addAttribute("reboardList", reboardList);

model.addAttribute("pagingInfo", paginationInfo);

return "/reboard/list";

}

```

@RequestMapping("/detail.do")
public String getReBoardDetail(int no, Model model, HttpServletRequest request){
    //상세보기 - no에 해당하는 글 조회
    //1. 파라미터
    logger.debug("파라미터 no={}", no);

    //2. db작업 - select
    ReBoardBean reboardBean = reBoardService.selectByNo(no);
    logger.debug("상세보기 조회 결과: reBoardBean={}", reboardBean);





    String content = reboardBean.getContent();
    //content 줄바꿈 처리
    if(content !=null && !content.isEmpty()){
        content = content.replace("\ r\ n", "<br>"); //엔터를 <br> 태그로 치환
        //하여 줄바꿈처리가 되도록 하자
    }
    reboardBean.setContent(content);

    //파일 첨부된 경우 파일정보와 다운로드 수 보여주기
    String fileName = reboardBean.getFileName();
    String fileInfo="", downInfo="";
    if(fileName!=null && !fileName.isEmpty()){
        fileInfo = "<img src=\"" +request.getContextPath()
            + "/images/file.gif" + "border='0'> "
            + reboardBean.getFileName() + " ("
            + (reboardBean.getFileSize()/1000) + " KB)";
        downInfo = "다운 : " + reboardBean.getDownCount();
    }
    //3. 결과, 뷰페이지 저장
    model.addAttribute("bean", reboardBean);
    model.addAttribute("fileInfo", fileInfo);
    model.addAttribute("downInfo", downInfo);
    return "/reboard/detail";
}

```

# 목록보기 페이지

## 자료실

번호	제목	작성자	작성일	조회수
10	 소스파일 첨부합니다 <small>NEW</small>	윤아	2013-03-10	13
9	 Re: 소스파일 첨부합니다 <small>NEW</small>	홍길동	2013-03-10	4
8	 안녕 <small>NEW</small>	홍길동	2013-03-09	11
7	 질문 <small>NEW</small>	김연아	2013-03-09	9
6	추천해주세요	손연재	2013-03-09	4

[1] [2]

제목

글쓰기

### ■ 자료실의 목록 보기

- 일반형 게시판 목록에서 첨부파일에 대한 부분만 추가
- 글이 파일을 첨부하고 있다는 의미로 파일이 첨부된 글은 글의 제목 앞에 간단한 이미지를 붙여 보여줌



# list.jsp

---

list.jsp

```
<!-- 자료실-첨부파일 있는 경우 파일 이미지 보여주기 -->
```

```
<c:if test="${!empty bean.fileName }">
```

```
    <img src = <c:url value="/images/file.gif"></c:url> border='0'>
```

```
</c:if>
```

# 상세보기 페이지

안녕

김길동

2014-11-04 17:59:04.0 조회수 : 11

한  
들

 [교재.txt \(2 KB\)](#) 다운 : 1

[수정](#) | [삭제](#) | [답변](#) | [목록](#)

- 자료실의 본문 내용보기
  - 첨부된 파일의 정보를 보여주고
  - 그 링크를 클릭하면 파일의 다운로드가 가능하게 하는 기능 추가



# ReboardController

//업로드된 파일정보 보여주기

@RequestMapping("/detail.do")

public String getReBoardDetail(int no, Model model, HttpServletRequest request){

.....  
//파일 첨부된 경우 파일정보와 다운로드 수 보여주기

String fileName = reboardBean.getFileName();

String fileInfo="", downInfo="";

if(fileName!=null && !fileName.isEmpty()){

    fileInfo = "<img src='" + request.getContextPath()

        + "/images/file.gif' " + "border='0'> "

        + reboardBean.getFileName() + " ("

        + (reboardBean.getFileSize()/1000)+" KB)";

    downInfo = "다운 : " + reboardBean.getDownCount();

}

//3. 결과, 뷰페이지 저장

model.addAttribute("bean", reboardBean);

model.addAttribute("fileInfo", fileInfo);

model.addAttribute("downInfo", downInfo);

return "/reboard/detail";

}



# detail.jsp

---

```
<div align="right">
    <a href='<c:url value
        ="/reboard/download.do?no=${bean.no}&fileName=${bean.fileName}" />'>
        ${fileInfo }
    </a>
    <span style="color:blue;font-weight:bold">
        ${downInfo }</span>
</div>

${bean.content}
```

# 파일 다운로드 처리

```
<beans:bean class="org.springframework.web.servlet.view.BeanNameViewResolver">
  <beans:property name="order" value="0" />
</beans:bean>
```

- property name="order" value="0" => 이렇게 설정을 해주어야 beanName이 String 타입으로 return 되었을때 이 BeanNameResolver가 받게 됨
- 우선순위를 0을 줌으로서 **BeanNameViewResolver**에서 **먼저 찾고** 없으면 InternalResourceViewResolver에서 찾게 됨

```
<beans:bean id="downloadView"
  class="com.herb.app.reboard.controller.ReBoardDownloadView" />
```

- bean을 하나 생성
  - 이 bean은 viewResolver를 통해서 파일 다운로드 처리를 하는 클래스로 넘어가게 됨
  - 이 클래스는 직접적으로 파일만 읽어와서 스트림으로 보내주는 작업을 함

```
@Component("downloadView")
public class ReBoardDownloadView extends AbstractView{
```





# 파일 다운로드 처리

- 1. 파일 다운로드 처리를 담당할 공통 클래스를 만들고 xml에 bean을 생성한다.
- 2. dispatcher-servlet.xml에 BeanNameViewResolver를 생성하고 order를 0으로 설정한다
- 3. Controller에서 return "생성한 빈네임" 으로 해야 하며 파일을 다운로드 할 객체를 model.addAttribute("")로 넘기거나 파일 path만 넘겨서 파일 객체를 얻어올 수 있어야 한다.

```
File myfile = new File(uploadPath, fileName);  
map.put("file", myfile);  
ModelAndView mav= new ModelAndView("downloadView", map);  
  
return mav;
```



# servlet-context.xml

---

```
<beans:bean
  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <beans:property name="prefix" value="/WEB-INF/views/" />
  <beans:property name="suffix" value=".jsp" />
  <beans:property name="order" value="1" />
</beans:bean>
```

```
<beans:bean class="org.springframework.web.servlet.view.BeanNameViewResolver">
  <beans:property name="order" value="0" />
</beans:bean>
```

```
<beans:bean id="downloadView"
  class="com.herb.app.reboard.controller.ReBoardDownloadView" />
```

```
@RequestMapping("/download.do")
```

```
public ModelAndView download(int no, String fileName, HttpServletRequest request){
```

```
    //db에서 다운로드 수 증가시키고, 다운로드 창을 띄우는 뷰 페이지로 넘긴다
```

```
    //1. 파라미터
```

```
    logger.debug("파라미터 : no={}, fileName={}", no, fileName);
```

```
    //2. db작업 - update
```

```
    int n= reBoardService.updateDownCount(no);
```

```
    logger.debug("다운로드 수 증가 결과: n={}", n);
```

```
    //3. 결과
```

```
    Map<String, Object> map = new HashMap<String, Object>();
```

```
    String uploadLastPath = fileuploadProperties.getProperty("file.upload.path");
```

```
    String savePath1 = request.getSession().getServletContext().getRealPath(uploadLastPath);
```

```
    String savePath = fileuploadProperties.getProperty("file.upload.path.test");
```

```
    logger.debug("savePath1 ={}, savePath={}", savePath1, savePath);
```

```
    File myfile = new File(savePath, fileName);
```

```
    map.put("file", myfile);
```

```
    ModelAndView mav = new ModelAndView("downloadView", map);
```

```
    return mav;
```

```
}
```

```
public ModelAndView(String viewName, Map<String, ?> model)  
public ModelAndView(View view, Map<String, ?> model)
```

```
public class ReboardDownloadView extends AbstractView {
```

```
    public ReboardDownloadView() {  
        setContentType("application/octet-stream");  
    }  
    @Override
```

ReboardDownloadView

강제 다운로드 창 띄우기

```
protected void renderMergedOutputModel(Map<String, Object> model,HttpServletRequest request,  
    HttpServletResponse response) throws Exception {  
    File file=(File)model.get("file");  
    if(file==null || !file.exists() || !file.canRead()){  
        response.setContentType("text/html;charset=utf-8");  
        PrintWriter out = response.getWriter();  
        out.println("<script>alert('파일이 존재하지 않거나 손상되었습니다');history.back();</script>");  
        return;  
    }  
}
```

```
System.out.println("파일명:"+file.getName());  
String fileName=new String(file.getName().getBytes("euc-kr"), "8859_1");  
  
response.setContentType(getContentType());  
response.setContentLength((int)file.length());  
response.setHeader("Content-disposition", "attachment;filename=" + fileName);  
  
response.setHeader("Content-Transfer-Encoding","binary");
```

```
<bean id="downloadView"  
    class="com.herb.app.reboard.controller.ReboardDownloadView" />
```

```
OutputStream os=response.getOutputStream();
FileInputStream fis=null;
try{
    fis=new FileInputStream(file);
    FileCopyUtils.copy(fis, os);
}finally{
    if(fis!=null){
        try{
            fis.close();
        }catch(IOException ex){
            ex.printStackTrace();
        }
    }
}
os.flush();
}

} //class
```



# 삭제하기

---

- 테이블의 데이터를 삭제하는 기능 이외에 첨부파일이 있는 경우 등록된 파일을 삭제하는 코드가 새롭게 추가

```

@RequestMapping(value="/delete.do", method=RequestMethod.POST)
public String deletePost(ReBoardBean bean, HttpServletRequest request, Model model) throws
    UnsupportedEncodingException{
    //db에서 삭제, 첨부된 file 삭제처리
    //1. 파라미터
    logger.debug("파라미터 bean={}", bean);

    //2. db작업 – delete
    //삭제 저장 프로시저 처리
    Map<String, String> map = new HashMap<String, String>();
    map.put("no", Integer.toString(bean.getNo()));
    map.put("step", Integer.toString(bean.getStep()));
    map.put("groupNo", Integer.toString(bean.getGroupNo()));

    String viewName="";
    if(reBoardService.checkPwd(bean.getNo(), bean.getPwd())){
        reBoardService.deleteReBoard(map);
        logger.debug("글 삭제");

        //파일 삭제
        String uploadLastPath = fileuploadProperties.getProperty("file.upload.path");
        String savePath1 = request.getSession().getServletContext().getRealPath(uploadLastPath);

        String savePath = fileuploadProperties.getProperty("file.upload.path.test");
        logger.debug("savePath1 = {}, savePath={}, savePath1, savePath);
    }
}

```

}



```

<mapper namespace="config.mybatis.mapper.oracle.reboard">
    <sql id="searchWhere">
        <where>
            <if test="searchKeyword !=null and searchKeyword !='' ">
                ${searchCondition} like '%'||#{searchKeyword}||'%'
            </if>
        </where>
    </sql>
    <insert id="reBoardInsert" parameterType="reBoardBean">
        <selectKey keyProperty="no" resultType="int" order="BEFORE">
            select reBoard_seq.nextval from dual
        </selectKey>
        insert into reBoard(no, name, pwd, title, email, content,
            groupNo, fileName, fileSize)
        values(#{no}, #{name}, #{pwd}, #{title}, #{email}, #{content},#{no}, #{fileName}, #{fileSize})
    </insert>
    <!-- 해당 페이지의 레코드만 조회하도록 변경 -->
    <select id="reBoardList" parameterType="searchBean" resultType="reBoardBean">
        SELECT *
        FROM (
            SELECT ROWNUM RNUM, ALL_LIST.*
            FROM (
                SELECT
                    no, name, pwd, title, email, content,regdate, readcount,
                    (sysdate-regdate)*24 as newlmgTerm, groupno, step, sortno,
                    delflag, filename, filesize, downcount
                FROM reBoard
            )
            <include refid="searchWhere"/>
    </select>

```

```

        order by groupNo desc, sortNo
    ) ALL_LIST
)
<![CDATA[
WHERE  RNUM > #{firstRecordIndex}
AND  RNUM <= #{firstRecordIndex} + #{recordCountPerPage} ]]>
</select>

<select id="getTotalRecord" parameterType="searchBean"    resultType="Integer">
    select count(*) from reboard
    <include refid="searchWhere"/>
</select>

<update id="reBoardUpdateReadCount" parameterType="int">
    update reBoard set readcount=readcount+1    where no=#{no}
</update>

<select id="getReBoardDetail" parameterType="int"    resultType="reBoardBean">
    select * from reBoard where no=#{no}
</select>

<select id="getReBoardPwd" parameterType="int" resultType="string">
    select pwd from reBoard where no=#{no}
</select>

<update id="reBoardUpdate" parameterType="reBoardBean">
    update reBoard set name=#{name}, title=#{title}, email=#{email}, content=#{content},
        fileName=#{fileName}, fileSize=#{fileSize}    where no=#{no}
</update>

```

```

<update id="reBoardUpdateDownCount" parameterType="int">
    update reBoard set downcount=downcount+1 where no=#{no}
</update>

<parameterMap type="map" id="reBoardDeleteParam">
    <parameter property="no" javaType="string" jdbcType="VARCHAR" mode="IN"/>
    <parameter property="step" javaType="string" jdbcType="VARCHAR" mode="IN"/>
    <parameter property="groupNo" javaType="string" jdbcType="VARCHAR" mode="IN"/>
</parameterMap>

<delete id="reBoardDeleteProc" parameterMap="reBoardDeleteParam">
    {call deleteReboard(?,?,?)}
</delete>

<insert id="reBoardReply" parameterType="reBoardBean">
    <selectKey keyProperty="no" resultType="int" order="BEFORE">
        select reBoard_seq.nextval from dual
    </selectKey>
    insert into reBoard(no, name, pwd, title, email, content, groupNo, step, sortNo)
    values("#{no}", #{name},#{pwd}, #{title}, #{email}, #{content},#{groupNo}, #{step}, #{sortNo})
</insert>

<update id="reBoardUpdateSortNo" parameterType="reBoardBean" >
    update reboard
    set sortno=sortno+1
    where groupNo=#{groupNo} and sortNo>#{sortNo}
</update>
</mapper>

```

## deleteReboard 저장 프로시저

```
create or replace procedure deleteReboard
(m_no number,
m_step number,
m_groupno number)
is
cnt number;
begin
  --원본글인 경우
  if m_step = 0 then
    --답변글이 존재하는지 체크
    select count(*) into cnt from reboard
    where groupNo = m_groupno;

    --답변글이 존재하는 경우
    if cnt > 1 then
      update reboard set DelFlag = 'Y'
      where no = m_no;
    else --답변글이 없는 경우
      delete reboard where no=m_no;
    end if;
  else --답변글 자체인 경우
    delete reboard where no=m_no;
  end if;

  commit;
```

```
EXCEPTION WHEN OTHERS THEN
  raise_application_error(-20001, ' 삭제 에러!');
  ROLLBACK;
END;
```

# 수정 페이지

## 글수정

제목 안녕

작성자 김길동

비밀번호

이메일 h1@naver.com

파일 첨부

찾아보기...

※ 첨부 파일을 새로 지정할 경우 기존 파일 📎 교재.txt은 삭제됩니다

내용

하나  
들

수정

글목록



# 수정 페이지

- 수정하기 페이지에서 새로 파일을 선택한 경우
  - 이전에 파일이 첨부된 경우엔 그 파일을 삭제
    - 이 파일이 존재하는가를 체크해서 존재한다면 삭제

```
//기존 파일은 삭제  
if (oldFile.exists()){  
    oldFile.delete();  
}
```

- 새로운 파일을 서버에 저장
- 새로 파일을 선택하지 않은 경우
  - 이전 파일명과 사이즈를 그대로 넘겨줌

# edit.jsp

```
<p>
  <label for="uploadFile">파일 첨부</label>
  <input type="file" name="uploadFile" id="uploadFile" size="45">
  <c:if test="${!empty bean.fileName && bean.fileName!='null' }">
    <br>
    <span style="color:darkgreen;font-weight:bold;font-size:0.8em">
      ※ 첨부 파일을 새로 지정할 경우 기존 파일
      <img src='<c:url value="/images/file.gif" />'>
      ${bean.fileName }은 삭제됩니다
    </span>
  </c:if>
</p>
```

```

@RequestMapping(value="/edit.do", method=RequestMethod.POST)
public String updatePost(ReBoardBean bean,
                        String oldFilename, String oldFilesize, Model model, HttpServletRequest request){
    //수정 처리 - update, 파일 업로드 처리
    //1. 파라미터
    logger.debug("파라미터 bean={}, oldfilename={}", bean, oldFilename);
    logger.debug("oldfilesize={}", oldFilesize);

    String fileName="";
    long fileSize=0;
    List<Map<String, Object>> resultList = null;
    try {
        resultList = reBoardService.fileupload(request);
        for(int i=0;i<resultList.size();i++){
            Map<String, Object> fileInfoMap = resultList.get(i);

            fileName = (String) fileInfoMap.get("fileName");
            fileSize =(Long) fileInfoMap.get("fileSize");
        }

        logger.debug("파일 업로드 성공, fileName={}, fileSize={}", fileName, fileSize);
    } catch (Exception e) {
        logger.debug("파일 업로드 실패");
        e.printStackTrace();
    }

    //새로 파일 첨부하는 경우 기존 파일 삭제
    if(resultList!=null && resultList.size()>0){
        //기존 파일 삭제
        String uploadLastPath = fileuploadProperties.getProperty("file.upload.path");
        String savePath1 = request.getSession().getServletContext().getRealPath(uploadLastPath);
        String savePath = fileuploadProperties.getProperty("file.upload.path.test");
        logger.debug("savePath1 ={}, savePath={}, savePath1, savePath);
    }
}

```



```

        if(oldFilename!=null && !oldFilename.isEmpty()){
            File oldFile = new File(savePath, oldFilename);
            if(oldFile.exists()){
                boolean flag = oldFile.delete();
                logger.debug("기존 파일 삭제 여부 : " + flag);
            }
        }
        bean.setFileName(fileName);
        bean.setFileSize(fileSize);
    }else{ //새로 파일 첨부하지 않는 경우
        bean.setFileName(oldFilename);
        bean.setFileSize(Integer.parseInt(oldFilesize));
    }

    //2. db작업 -update
    String msg="", url="";
    if(reBoardService.checkPwd(bean.getNo(), bean.getPwd())){
        //비밀번호가 일치하는 경우 수정처리
        int n = reBoardService.updateReBoard(bean);
        if(n>0){//성공
            url="detail.do?no="+bean.getNo();
        }else{//실패
            msg="글 수정 실패!";
            url="edit.do?no="+bean.getNo();
        }
    }else{
        msg="비밀번호가 일치하지 않습니다!";
        url="edit.do?no="+bean.getNo();
    }

    //3.결과, 뷰페이지
    model.addAttribute("msg", msg);
    model.addAttribute("url", url);
    return "/inc/message";
}

```



## 답변형 게시판

---

## 답변형 게시판

번호	제목	작성자	작성일	조회수
9	책추천 <b>NEW</b>	유재석	2013-03-06	0
8	 Re:책추천할께요 <b>NEW</b>	송혜교	2013-03-06	0
7	 Re:Re:책추천할께요 <b>NEW</b>	윤아	2013-03-06	0
6	 Re:책추천 <b>NEW</b>	조인성	2013-03-06	0
5	반가워 <b>NEW</b>	손연재	2013-03-06	0

[1] [2] [3]

제목 

글쓰기

# detail.jsp

안녕

김길동

2014-11-04 17:59:04.0 조회수 : 11

📎 교재.txt (2 KB) 다운 : 1

한  
들  
서이

수정 | 삭제 | **답변** | 목록

```
<a href='<c:url value="/reboard/edit.do?no=${bean.no}"></c:url>'>수정</a> |  
<a href='<c:url value  
="/reboard/delete.do?no=${bean.no}&fileName=${bean.fileName }&step=${bean.step}&groupNo  
=${bean.groupNo }" />' > 삭제</a> |  
<a href='<c:url value ="/reboard/reply.do?no=${bean.no}" />' >답변</a> |  
<a href='<c:url value="/reboard/list.do" />' >목록</a>
```



# reply.jsp

---

## 답변하기

제목 Re : 안녕

작성자

비밀번호

이메일

내용

등록

글목록



# 답변하기 - reply.jsp

---

## ■ 글쓰기(답변하기)

- 새로운 글을 올릴 때는 write.jsp, 어떤 글의 답변을 할 경우에는 reply.jsp 사용
- 새 글
  - 글이 저장될 때 GroupNo는 no로 저장, step, SortNo은 0으로 세팅
- 답변하기
  - 원본글의 no로 GroupNo, step, SortNo를 조회해와서
  - DB에 글을 저장할 때 GroupNo 는 그대로,
  - step와 SortNo는 +1 을 해서 값을 넣어줌
- View.jsp에서 넘어온 no값을 가지고 원래 글의 제목과 내용을 DB에서 조회해와서 미리 보여줌.
- 제목에는 Re :



# ReBoardDAOMybatis

```
public int updateSortNo(ReBoardBean bean){
    int n = getSqlSession().update(namespace+".reBoardUpdateSortNo", bean);
    return n;
}
public int replyReBoard(ReBoardBean bean){
    bean.setStep(bean.getStep()+1);
    bean.setSortNo(bean.getSortNo()+1);

    int n = (Integer) getSqlSession().insert(namespace+".reBoardReply", bean);
    return n;
}
```

-----  
[ ReboardService ]

```
@Transactional
public int replyReBoard(ReBoardBean bean){
    //서비스에서 트랜잭션 처리
    int n = boardDao.updateSortNo(bean);
    n = boardDao.replyReBoard(bean);

    return n;
}
```



# 계층적으로 리스트 보여주기

```
select * from Reboard  
order by groupNo desc, sortNo asc
```

- list.jsp

```
<!-- 답변형 게시판에서 답변인 경우 단계별로 보여주기 -->  
<c:if test="${vo.step>0}">  
    <c:forEach var="i" begin="1" end="${vo.step*2}">  
        &nbsp; &nbsp;  
    </c:forEach>  
      
</c:if>
```



## 답변형 게시판

번호	제목	작성자	작성일	조회수
9	책추천 <b>NEW</b>	유재석	2013-03-06	0
8	 Re:책추천할께요 <b>NEW</b>	송혜교	2013-03-06	0
7	 Re:Re:책추천할께요 <b>NEW</b>	윤아	2013-03-06	0
6	 Re:책추천 <b>NEW</b>	조인성	2013-03-06	0
5	반가워 <b>NEW</b>	손연재	2013-03-06	0

[1] [2] [3]

제목 

글쓰기



## 삭제하기-Flag처리

---

- reboard table의 DelFlag 컬럼 처리
  - 1. 답변이 있는 원본글인 경우에는 레코드를 삭제하지 말고 DelFlag = “Y” 로 update 한다
  - 2. list 페이지에서, DelFlag를 조회해 와서 값이 “Y”이면 제목에 링크 걸지 말고, font color를 회색으로 보여준다.

# 삭제하기-Flag처리

## 답변형 게시판

번호	제목	작성자	작성일	조회수
14	안녕 <span>NEW</span>	홍길동	2013-03-07	2
13	질문	김길동	2013-03-06	2
12	Re:질문	질문자	2013-03-06	5
11	Re:Re:질문	손연재	2013-03-06	2
10	테스트	김연아	2013-03-06	1

[1] [2] [3]

제목

검색

글쓰기

## list.jsp

```
<!-- 답변형 게시판 - 답변이 있는 원본글이 삭제처리(delFlag='Y')된 경우 -->
<c:if test="${bean.delFlag == 'Y'}"> <!-- 제목을 회색으로 보여주고 링크를 걸지 않는다 -->
    <font color='#C2C2C2'>삭제된 글입니다</font>
</c:if>
<c:if test="${bean.delFlag != 'Y'}"> <!-- 삭제처리 안된 글 -> 제목에 링크 걸고, 검정색으로 보여준다 -->

    <!--제목이 30자 이상인 경우 처리 -->
    <a href='<c:url value="/reboard/countUpdate.do?no=${bean.no}"></c:url>'>
        <c:if test="${fn:length(bean.title)>30}">
            ${fn:substring(bean.title, 0, 30)}....
        </c:if>
        <c:if test="${fn:length(bean.title)<=30}">
            ${bean.title}
        </c:if>
    </a>

    <!-- new 이미지 처리 -->
    <c:if test="${bean.newImgTerm <= 24}">
        <img src='<c:url value="/images/new.gif"></c:url>' border="0">
    </c:if>
</c:if>
```

```

<!-- 삭제된 글인 경우 링크걸지 말고, 회색으로 보여준다 -->
<c:if test="${fn:toUpperCase(vo.delFlag)=='Y'}">
    <span style="color:gray">
        삭제된 글입니다.
    </span>
</c:if>
<c:if test="${fn:toUpperCase(vo.delFlag)!='Y'}">
    <!-- 답변형 게시판에서 답변인 경우 단계별로 보여주기 -->

    <c:if test="${vo.step>0}">
        <c:forEach var="i" begin="1"
            end="${vo.step*2}">
            &nbsp;
        </c:forEach>
        
            alt="답변이미지" />
    </c:if>

    <!-- 파일이 첨부된 경우 파일 이미지 보여주기 -->
    <c:if test="${!empty vo.fileName }">
        
    </c:if>

    <a href
    ="<c:url value='/reBoard/countUpdate.do?no=${vo.no}'/">">

```

```

<!-- 제목의 길이가 30자 이상인 경우 30자만 보여주기 -->
    <c:if test="${fn:length(vo.title)>30}">
        ${fn:substring(vo.title, 0, 30)}...
    </c:if>
    <c:if test="${fn:length(vo.title)<=30}">
        ${vo.title}
    </c:if>
</a>

<!-- 24시간 이내의 글인 경우 new 이미지 보여주기 -->
<c:if test="${vo.dateTerm<24}">
    
        alt="new 이미지" />
</c:if>
</c:if>

```

# 게시판 검색기능

select \* from reboard where name like '%길동%'

select \* from reboard where title like '%안녕%'

select \* from reboard where content like '%질문%'

## 답변형 게시판

검색어 : 길동, 3건이 검색되었습니다.

번호	제목	작성자	작성일	조회수
3	안녕 <small>NEW</small>	홍길동	2013-03-07	2
2	질문	김길동	2013-03-06	2
1	안녕	홍길동	2013-03-06	2

[1]

작성자 ▼ 길동 검색

글쓰기

```
@RequestMapping(value="/reply.do", method=RequestMethod.GET)
public String replyGet(@RequestParam("no") int no, Model model){
    //답변달기 화면 보여주기
    //1. 파라미터
    logger.debug("파라미터, no={}", no);

    //2. db작업-select
    ReBoardBean bean=reBoardService.selectByNo(no);

    //3. 결과 저장, 리턴
    model.addAttribute("bean", bean);
    return "/reboard/reply";
}

@RequestMapping(value="/reply.do", method=RequestMethod.POST)
public String replyPost(ReBoardBean bean){
    //답변달기 처리
    //1. 파라미터
    logger.debug("파라미터, bean={}", bean);

    //2. db작업 - insert
    int key= reBoardService.replyReBoard(bean);
    logger.debug("답변달기 결과, key={}", key);

    //3. 결과 저장, 리턴
    return "redirect:list.do";
}
```