

코드로 배우는 스프링 웹 프로젝트

• PART 6



02 파일의 확장자나 크기 사전처리

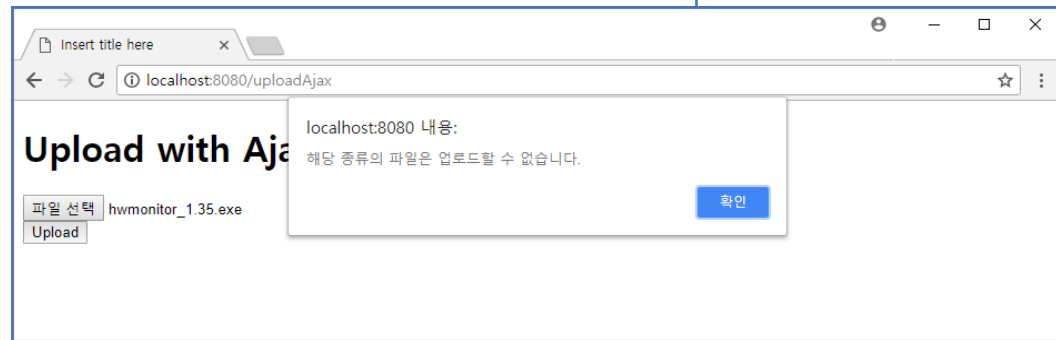
- 정규식을 이용해서 파일 확장자 체크

```
var regex = new RegExp("(.*?)\\.(exe|sh|zip|alz)$");
var maxSize = 5242880; //5MB

function checkExtension(fileName, fileSize){

    if(fileSize >= maxSize){
        alert("파일 사이즈 초과");
        return false;
    }

    if(regex.test(fileName)){
        alert("해당 종류의 파일은 업로드할 수 없습니다.");
        return false;
    }
    return true;
}
```



02 중복된 이름의 파일 처리

- UUID를 이용해서 고유한 문자열 생성후 업로드된 파일의 이름으로 사용
- 하나의 폴더에 너무 많은 파일이 업로드 되지 않도록 '년/월/일' 폴더 생성

```
private String getFolder() {  
  
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");  
  
    Date date = new Date();  
  
    String str = sdf.format(date);  
  
    return str.replace("-", File.separator);  
}
```

```
// make folder -----  
File uploadPath = new File(uploadFolder, getFolder());  
log.info("upload path: " + uploadPath);
```

```

UUID uuid = UUID.randomUUID();

uploadFileName = uuid.toString() + "_" + uploadFileName;

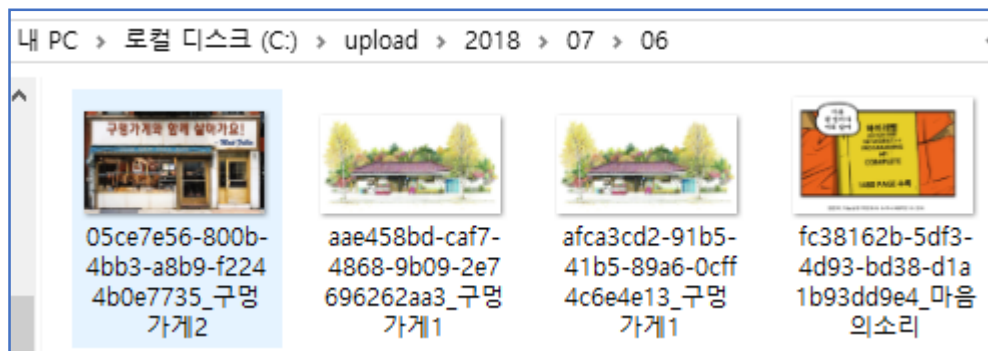
File saveFile = new File(uploadPath, uploadFileName);

try {

    multipartFile.transferTo(saveFile);
} catch (Exception e) {
    Log.error(e.getMessage());
} // end catch

```

UUID를 이용해서 고유한 파일명 생성



02 섬네일(thumbnail) 이미지 생성

- 원본 이미지를 그냥 사용하는 경우 브라우저에 너무 많은 데이터들이 전송되는 문제
- 일반적으로 작은 이미지(섬네일)를 생성해서 처리
- JDK의 ImageIO를 이용할 수도 있지만, 해상도 등의 문제로 인해 별도의 라이브러리 활용
 - Thumbnailator 라이브러리 활용

Home » net.coobird » thumbnailator

 **Thumbnailator**
Thumbnailator - a thumbnail generation library for Java

License: MIT

Used By: 48 artifacts

Central (8) Jahia (1)

Version
0.4.8
0.4.7
0.4.6

```
<dependency>  
  <groupId>net.coobird</groupId>  
  <artifactId>thumbnailator</artifactId>  
  <version>0.4.8</version>  
</dependency>
```

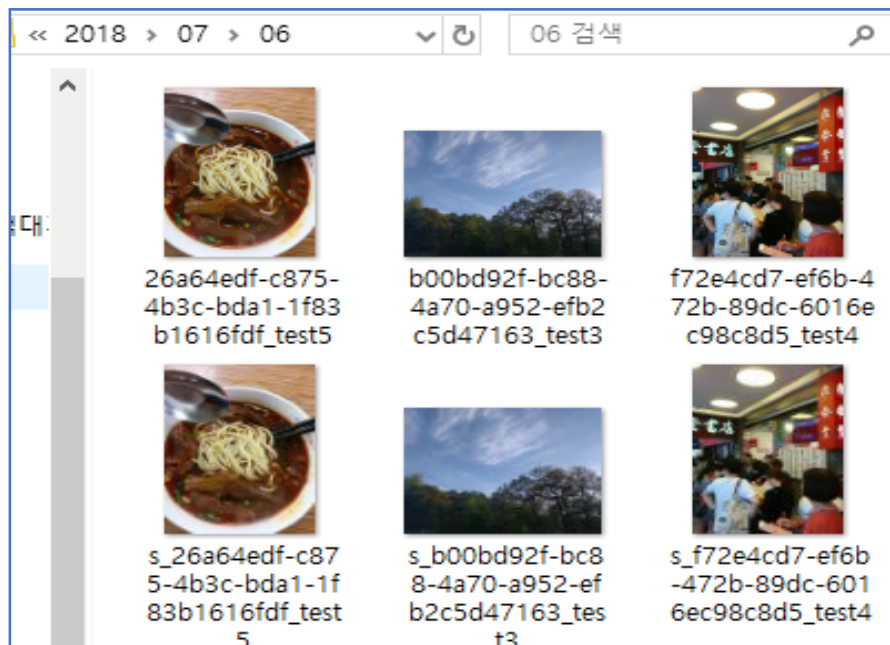
02 섬네일을 처리하는 단계

- 이미지 파일 여부의 판단
- 섬네일 생성 및 저장

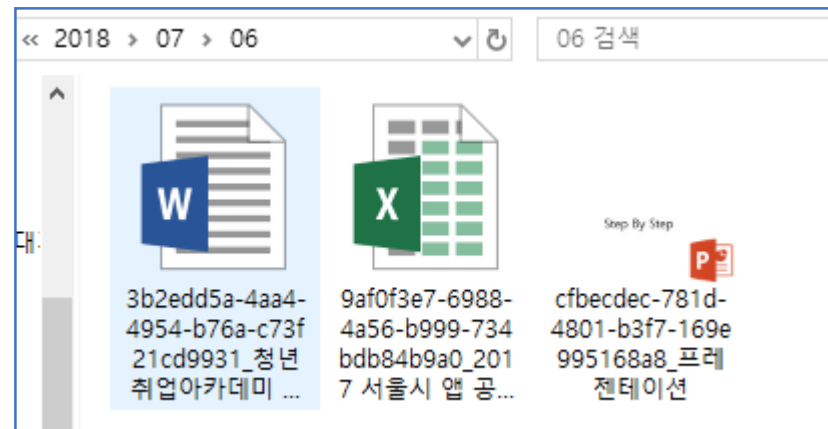
```
try {  
    String contentType = Files.probeContentType(file.toPath());  
  
    return contentType.startsWith("image");  
  
} catch (IOException e) {  
    // TODO Auto-generated catch block  
    e.printStackTrace();  
}
```

```
// check image type file  
  
if (checkImageType(saveFile)) {  
  
    FileOutputStream thumbnail =  
        new FileOutputStream(new File(uploadPath, "s_" + uploadFileName));  
  
    Thumbnailator.createThumbnail(multipartFile.getInputStream(), thumbnail, 100, 100);  
  
    thumbnail.close();  
}
```

이미지파일의 경우



일반파일의 경우



02 업로드된 파일의 데이터 반환

- 업로드 이후에 반환해야 하는 정보
 - 업로드된 파일의 이름과 원본 파일의 이름
 - 파일이 저장된 경로
 - 업로드된 파일이 이미지인지 아닌지에 대한 정보
- 정보들을 객체로 처리하고 JSON으로 전송

```
@Data
public class AttachFileDTO {

    private String fileName;
    private String uploadPath;
    private String uuid;
    private boolean image;

}
```


02 브라우저에서 Ajax의 처리

```
$.ajax({  
  url: '/uploadAjaxAction',  
  processData: false,  
  contentType: false,  
  data: formData,  
  type: 'POST',  
  dataType: 'json',  
  success: function(result){  
  
    console.log(result);  
  
  }  
}); //$.ajax
```

```
▼ (2) [{...}, {...}] ⓘ  
  ▼ 0:  
    fileName: "test4.jpg"  
    image: true  
    uploadPath: "2018\07\06"  
    uuid: "5ec7b354-690f-48f9-ae88-8fd6a6061ccf"  
    ▶ __proto__: Object  
  ▼ 1:  
    fileName: "test5.jpg"  
    image: true  
    uploadPath: "2018\07\06"  
    uuid: "1586d515-2921-4c43-8a8f-3c1a1f600e41"  
    ▶ __proto__: Object  
  length: 2  
  ▶ __proto__: Array(0)
```

```
$.ajax({
    url : '/uploadAjaxAction',
    processData : false,
    contentType : false,
    data : formData,
    type : 'POST',
    dataType : 'json',
    success : function(result) {

        console.log(result);

        showUploadedFile(result);

        $(".uploadDiv").html(cloneObj.html());

    }
}); //$.ajax
```

```
function showUploadedFile(uploadResultArr) {

    var str = "";

    $(uploadResultArr).each(function(i, obj) {

        str += "<li>" + obj.fileName + "</li>";

    });

    uploadResult.append(str);
}
```

Upload with Ajax

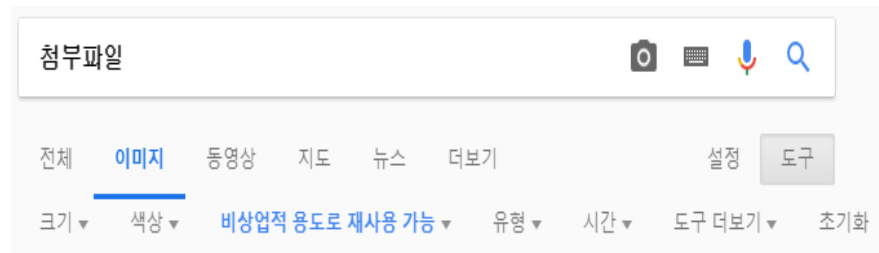
파일 선택 선택된 파일 없음

Upload

- test4.jpg
- test4.jpg
- test5.jpg

02 일반 파일의 처리

- 이미지가 아닌 경우에는 화면에서는 간단한 아이콘 등을 이용해서 첨부파일 표시



Upload with Ajax

파일 선택 선택된 파일 없음

📎그림파트2.pptx 📎그림파트3.pptx 📎그림파트4.pptx

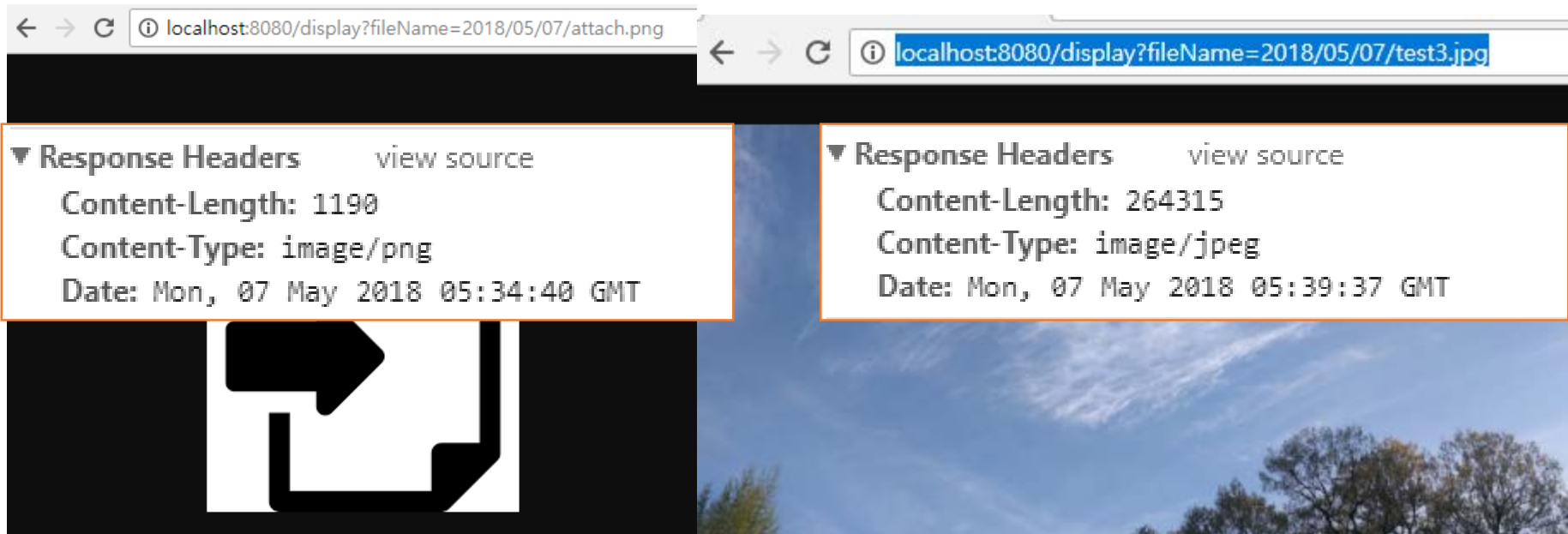
Upload

02 섬네일 이미지 보여주기

- 파일의 확장자에 따라 적당한 MIME타입 데이터를 지정

```
HttpHeaders header = new HttpHeaders();  
  
header.add("Content-Type", Files.probeContentType(file.toPath()));
```

파일의 확장자에 따라서 MIME타입이 다르게 처리



- 일반 파일의 경우 클릭시에 다운로드 처리

```
@GetMapping(value = "/download", produces = MediaType.APPLICATION_OCTET_STREAM_VALUE)
@ResponseBody
public ResponseEntity<Resource> downloadFile(String fileName) {

    Log.info("download file: " + fileName);

    Resource resource = new FileSystemResource("c:\\upload\\" + fileName);

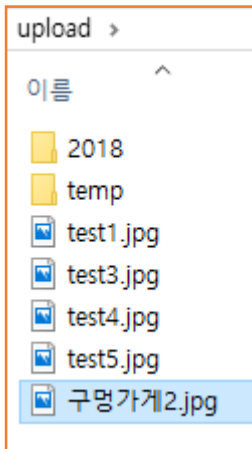
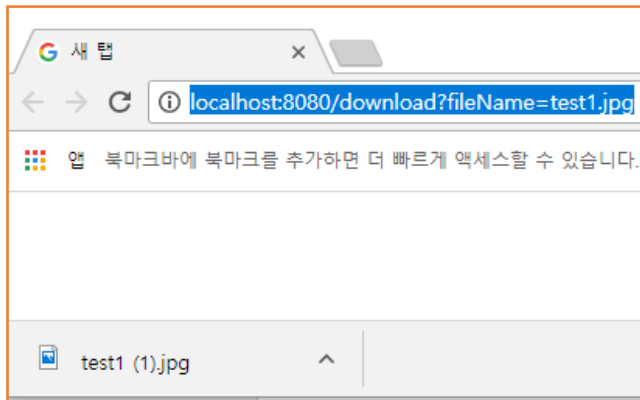
    Log.info("resource: " + resource);

    String resourceName = resource.getFilename();
    HttpHeaders headers = new HttpHeaders();

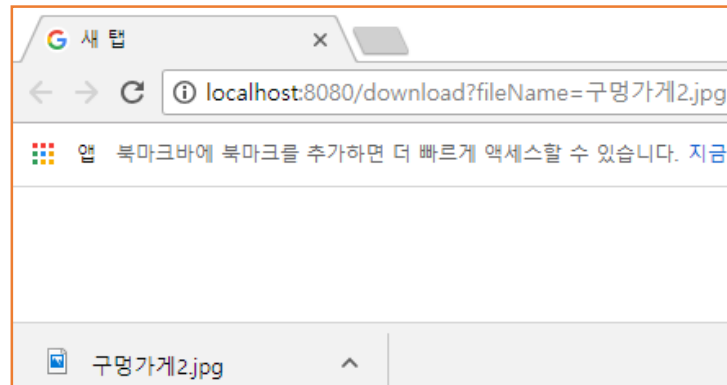
    try {
        headers.add("Content-Disposition",
            "attachment; filename=" + new String(resourceName.getBytes("UTF-8"), "ISO-8859-1"));
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }

    return new ResponseEntity<Resource>(resource, headers, HttpStatus.OK);
}
```

자동으로 해당 파일을 다운로드



한글이름 파일 다운로드



IE의 경우에 한글 파일 이름 처리 주의

```

boolean checkIE = (userAgent.indexOf("MSIE") > -1 || userAgent.indexOf("Trident") > -1);

String downloadName = null;

if (checkIE) {
    downloadName = URLEncoder.encode(resourceName, "UTF8").replaceAll("\\+", " ");
} else {
    downloadName = new String(resourceName.getBytes("UTF-8"), "ISO-8859-1");
}

```

02 화면에서의 다운로드 처리

- 화면에서는 '/download' 경로로 요청

```
if(!obj.image){  
  
    var fileCallPath = encodeURIComponent( obj.uploadPath+"/"+ obj.uuid + "_" +obj.fileName);  
  
    str += "<li><a href='/download?fileName="+fileCallPath+"'>  
        +<img src='/resources/img/attach.png'" +obj.fileName+"</a></li>"
```



02 원본 이미지 보여주기

- <div>를 이용해서 화면 내에 원본 이미지를 보여주도록 처리
- jQuery의 animate()를 이용해서 처리

```
$(".bigPictureWrapper").css("display", "flex").show();  
  
$(".bigPicture")  
  .html("<img src='/display?fileName="+fileCallPath+"'>")  
  .animate({width: '100%', height: '100%'}, 1000);
```

Upload with Ajax

파일 선택 선택된 파일 없음
Upload



02 첨부파일의 삭제

- 첨부파일 삭제시 고려해야 하는 사항들
 - 이미지 파일의 경우에는 섬네일까지 같이 삭제되어야 하는 점
 - 파일을 삭제한 후에는 브라우저에서도 섬네일이나 파일 아이콘이 삭제되도록 처리하는 점
 - 비정상적으로 브라우저의 종료 시 업로드된 파일의 처리
- 화면에서의 삭제
 - 첨부파일의 <div> 자체를 삭제
- 서버에서의 삭제
 - 이미지의 경우에는 섬네일 삭제/ 원본 삭제
 - 일반 파일의 경우에는 원본 삭제

- 첨부파일은 삭제했지만, 원래의 게시글을 수정/삭제하지 않은 경우
- 비정상적으로 수정/삭제 중에 브라우저가 종료된 경우
- 무난한 방식은 DB에 있는 첨부파일의 목록과 실제 업로드 폴더에 있는 파일의 목록을 비교해서 처리하는 작업을 주기적으로 처리
- Quartz라이브러리를 이용해서 스프링에서 주기적으로 처리

02 등록을 위한 화면 처리

- 추가된 첨부파일은 이미 업로드가 완료된 상황
- <form> 전송시 <input type='hidden'> 태그들을 첨부된 파일의 수 만큼 생성해서 같이 전송

JSON정보는 <input type='hidden'>으로 변환

```
<input type="hidden" name="attachList[0].fileName" value="test3.jpg">
<input type="hidden" name="attachList[0].uuid" value="cf4a5227-e9c5-460f-b8f9-1e9f672a330b">
<input type="hidden" name="attachList[0].uploadPath" value="2018\05\16">
<input type="hidden" name="attachList[0].fileType" value="true">
<input type="hidden" name="attachList[1].fileName" value="test4.jpg">
<input type="hidden" name="attachList[1].uuid" value="94c87808-604c-48b4-8b34-b35ee7af1036">
<input type="hidden" name="attachList[1].uploadPath" value="2018\05\16">
<input type="hidden" name="attachList[1].fileType" value="true">
<input type="hidden" name="attachList[2].fileName" value="test5.jpg">
<input type="hidden" name="attachList[2].uuid" value="c4b0c2ac-abdd-49f3-890f-c7ba1a24b3cc">
<input type="hidden" name="attachList[2].uploadPath" value="2018\05\16">
<input type="hidden" name="attachList[2].fileType" value="true">
```

```
create table tbl_attach (  
  uuid varchar2(100) not null,  
  uploadPath varchar2(200) not null,  
  fileName varchar2(100) not null,  
  filetype char(1) default 'I',  
  bno number(10,0)  
);  
  
alter table tbl_attach add constraint pk_attach primary key (uuid);  
  
alter table tbl_attach add constraint fk_board_attach foreign key (bno)  
references tbl_board(bno);
```

tbl_board와는 외래키로 설정

```
@Data
public class BoardAttachVO {

    private String uuid;
    private String uploadPath;
    private String fileName;
    private boolean fileType;

    private Long bno;

}
```

<input type='hidden'>으로 만들어진
파일 정보들을 BoardAttachVO로 변환

```
public class BoardVO {

    private Long bno;
    private String title;
    private String content;
    private String writer;
    private Date regdate;
    private Date updateDate;

    private int replyCnt;

    private List<BoardAttachVO> attachList;

}
```

BoardVO는 여러 개의 첨부 파일을 가지도록

```
@Transactional
```

```
@Override
```

```
public void register(BoardVO board) {
```

```
    Log.info("register....." + board);
```

```
    mapper.insertSelectKey(board);
```

```
    if(board.getAttachList() == null || board.getAttachList().size() <= 0) {
```

```
        return;
```

```
    }
```

```
    board.getAttachList().forEach(attach ->{
```

```
        attach.setBno(board.getBno());
```

```
        attachMapper.insert(attach);
```

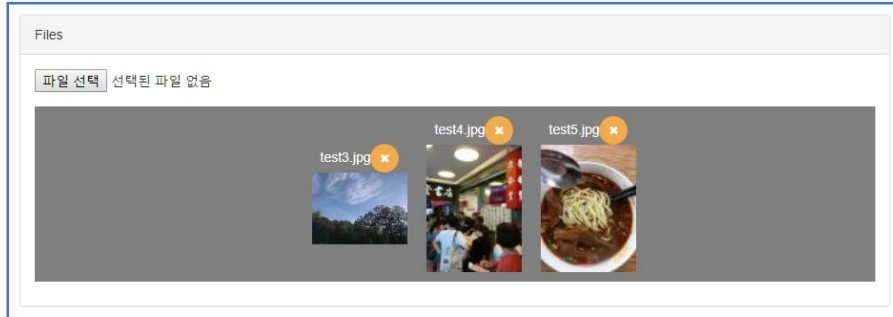
```
    });
```

```
}
```

트랜잭션하에 여러 개의 첨부 파일 정보도
DB에 저장

test3.jpg, test4.jpg, test5.jpg 를 게시물 등록 시에 추가한 경우

UUID	UPLOADPATH	FILENAME	FILETYPE	BNO
1 e7f922a7-1456-40fa-85ae-2dabb9f85b73	2018\07\08	test3.jpg	1	3145784
2 9fa37bc2-415f-4551-9fb2-97892e219f02	2018\07\08	test5.jpg	1	3145784
3 d4f32238-7f00-4a50-b4e6-2d1441501e62	2018\07\08	test4.jpg	1	3145784



02 게시물의 조회와 첨부파일

- 게시물 조회시에는 게시물의 정보와 첨부파일들의 정보를 같이 가져오도록 해야 함
 - Ajax로 해당 게시물의 첨부파일들만 조회하는 방식

```
@GetMapping(value = "/getAttachList", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
@ResponseBody
public ResponseEntity<List<BoardAttachVO>> getAttachList(Long bno) {

    Log.info("getAttachList " + bno);

    return new ResponseEntity<>(service.getAttachList(bno), HttpStatus.OK);

}
```

```
var bno = '<c:out value="${board.bno}"/>';

$.getJSON("/board/getAttachList", {bno: bno}, function(arr){

    console.log(arr);

}); //end getjson
```


02 게시물의 삭제와 첨부파일

- 게시물의 삭제시에는 데이터베이스 상의 파일 정보의 삭제
- 실제 업로드된 파일 (이미지의 경우 썸네일도) 삭제

게시물 수정 삭제

Board Modify Page

Board Modify Page

Bno
2097170

Title
Remove Test

Text area
Remove Test

Writer
writer

데이터베이스내 첨부 파일의 기록

UUID	UPLOAD...	FILENAME	FILETYPE	BNO

폴더내 썸네일과 일반 파일 삭제



02 서버측 게시물 수정과 첨부파일의 삭제

- 첨부파일은 사실상 수정이라는 개념이 존재하지 않음
- 삭제후 다시 추가하는 방식

test3.jpg, test4.jpg, test5.jpg 를 게시물 등록 시에 추가한 경우

Files

파일 선택 선택된 파일 없음

test3.jpg

test4.jpg

test5.jpg

UPLOADPATH	FILENAME	FILETYPE	BNO
2018\07\08	test3.jpg	1	3145784
2018\07\08	test5.jpg	1	3145784
2018\07\08	test4.jpg	1	3145784

test3.jpg, test4.jpg를 삭제하고 jeju.jpg파일을 추가한 경우



Files

파일 선택 jeju.jpg

test5.jpg

jeju.jpg

UUID	UPLOADPATH	FILENAME	FILETYPE	BNO
1 9fa37bc2-415f-4551-9fb2-97892e219f02	2018\07\08	test5.jpg	1	3145784
2 7fe55ea1-8688-4f90-9625-b3db746aa9aa	2018\07\08	jeju.jpg	1	3145784

02 잘못된 첨부파일들의 삭제

- 첨부파일정보와 DB상의 정보가 일치하지 않는 상황
 - 첨부파일만을 등록하고 게시물을 등록하지 않았을 때의 문제 - 파일은 이미 서버에 업로드되었지만, 게시물을 등록하지 않았으므로 의미 없이 파일들만 서버에 업로드된 상황
 - 게시물을 수정할 때 파일을 삭제했지만 실제로 폴더에서 기존 파일은 삭제되지 않은 문제
 - 데이터베이스에는 기존 파일이 삭제되었지만, 실제 폴더에는 남는 문제

어제 날짜로 등록된 첨부파일의 목록을 구한다.



어제 업로드가 되었지만, 데이터베이스에는 존재하지 않는 파일들을 찾는다.



데이터베이스와 비교해서 필요 없는 파일들을 삭제한다.

UUID	UPLOADPATH	FILENAME	FILETYPE
e941b1c6-6fec-458...	2018W05W21	test3.jpg	1
e7c71a10-7aab-4a...	2018W05W21	test4.jpg	1
b4d2bd62-03bf-45...	2018W05W21	test5.jpg	1
2726675a-c695-4d...	2018W05W21	0c2229a.jpg	1
20a6b192-3bad-43...	2018W05W21	888627.jpg	1
97a57a7b-a982-4f...	2018W05W21	c0022996_491ae4059d9...	1
e8f3ed3d-43e5-45...	2018W05W21	freesoul_76669_1[9].jpg	1
758f8d17-e89b-42...	2018W05W21	jeju.jpg	1
212aba25-e5b0-49...	2018W05W21	z7676.jpg	1
ba6c8831-80ba-4fa...	2018W05W21	z7677.jpg	1

2f51c91d-6a26-48d4-a713-9e33929902b1_20180318
 3c057106-d2f3-4add-8647-0b1c4c045738_NCS평가양식0222
 20a6b192-3bad-4315-8962-9738c9db4dd4_888627
 97a57a7b-a982-4f7e-a938-aca1af338a40_c0022996_491ae4059d958
 212aba25-e5b0-494e-bccc-2cc850a98a03_z7676
 758f8d17-e89b-42d8-b5a6-15cd0cb01bff_jeju
 1069d80b-c4bb-47a6-af97-5ed20482f431_프레젠테이션
 2726675a-c695-4d51-8a1c-d8c6d29d83b2_0c2229a
 b4d2bd62-03bf-45e4-b23c-ff698c440c34_test5
 ba6c8831-80ba-4fa9-bd92-43753a9c94f8_z7677
 bc96f685-5598-48f6-b1ea-67b884d9aa29_NCS평가양식0226
 bd01b459-9f14-44b5-839e-ae16b988304f_c0022996_491ae4059d9...
 e7c71a10-7aab-4a45-9517-05b2208fb6d0_test4
 e8f3ed3d-43e5-4542-9453-73e4a8ab7007_freesoul_76669_1[9]
 e941b1c6-6fec-4589-9c11-35c365026e91_test3
 s_20a6b192-3bad-4315-8962-9738c9db4dd4_888627
 s_97a57a7b-a982-4f7e-a938-aca1af338a40_c0022996_491ae4059d9...
 s_212aba25-e5b0-494e-bccc-2cc850a98a03_z7676
 s_758f8d17-e89b-42d8-b5a6-15cd0cb01bff_jeju

20a6b192-3bad-4315-8962-9738c9db4dd4_888627
 97a57a7b-a982-4f7e-a938-aca1af338a40_c0022996_491ae4059d958
 212aba25-e5b0-494e-bccc-2cc850a98a03_z7676
 758f8d17-e89b-42d8-b5a6-15cd0cb01bff_jeju
 2726675a-c695-4d51-8a1c-d8c6d29d83b2_0c2229a
 b4d2bd62-03bf-45e4-b23c-ff698c440c34_test5
 ba6c8831-80ba-4fa9-bd92-43753a9c94f8_z7677
 e7c71a10-7aab-4a45-9517-05b2208fb6d0_test4
 e8f3ed3d-43e5-4542-9453-73e4a8ab7007_freesoul_76669_1[9]
 e941b1c6-6fec-4589-9c11-35c365026e91_test3
 s_20a6b192-3bad-4315-8962-9738c9db4dd4_888627
 s_97a57a7b-a982-4f7e-a938-aca1af338a40_c0022996_491ae4059d9...
 s_212aba25-e5b0-494e-bccc-2cc850a98a03_z7676
 s_758f8d17-e89b-42d8-b5a6-15cd0cb01bff_jeju
 s_2726675a-c695-4d51-8a1c-d8c6d29d83b2_0c2229a
 s_b4d2bd62-03bf-45e4-b23c-ff698c440c34_test5
 s_ba6c8831-80ba-4fa9-bd92-43753a9c94f8_z7677
 s_e7c71a10-7aab-4a45-9517-05b2208fb6d0_test4
 s_e8f3ed3d-43e5-4542-9453-73e4a8ab7007_freesoul_76669_1[9]
 s_e941b1c6-6fec-4589-9c11-35c365026e91_test3

02 Quartz라이브러리 설정

- 주기적으로 반복해야 하는 작업을 Java를 이용해서 처리할 수 있도록 하는 경우에 사용

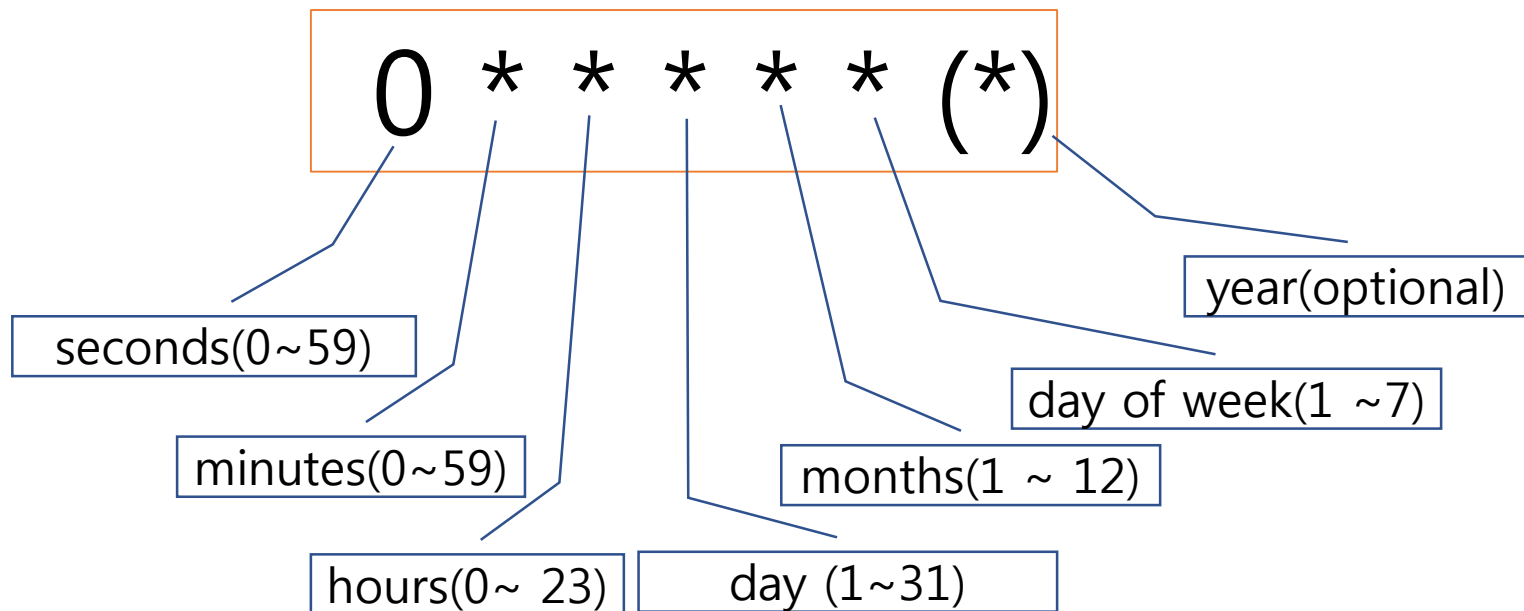
```
<!-- https://mvnrepository.com/artifact/org.quartz-scheduler/quartz -->
<dependency>
  <groupId>org.quartz-scheduler</groupId>
  <artifactId>quartz</artifactId>
  <version>2.3.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.quartz-scheduler/quartz-jobs -->
<dependency>
  <groupId>org.quartz-scheduler</groupId>
  <artifactId>quartz-jobs</artifactId>
  <version>2.3.0</version>
</dependency>
```

- root-context.xml에 설정 추가

```
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring
http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-4.3.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
...생략...
<task:annotation-driven/>
```

- 시간 단위의 값을 조정해서 주기 설정



열공합시다!!

