Gallery
- 파일 업로드 -

제목

배트맨

설명

배트맨 오토바이트

이미지 파일들

파일 선택 batman.jpg

확인 돌아가기

❖ gallery/upload.jsp

```
<div style="width:70%" class="mt-4 mb-3 ml-auto mr-auto">
   <h1>이미지 업로드</h1>
   <form:form commandName="image" enctype="multipart/form-data">
      <div class="md-form">
         <label for="title">제목</label>
         <form:input path="title" required="required"/>
         <form:errors path="title"/>
      </div>
      <div>
         <label for="description">설명</label>
         <form:textarea path="description" rows="4" />
      </div>
      <div >
         <label>이미지 파일들</label>
      </div>
      <div>
         <input type="file" name="imageFiles"
              multiple="multiple" accept="image/*" required>
      </div>
```

❖ gallery/upload.jsp

```
      <div class="text-center">
         <button type="submit" class="btn btn-primary btn-md">
            <i class="fa fa-check-circle mr-2"></i> 확인</button>
         </button>
         <a href="javascript:history.back()"
            class="btn btn-primary btn-md">돌아가기</a>
      </div>
   </form:form>
</div>
```

파일 업로드

❖ 폼 인코딩

- enctype="application/x-www-form-urlencoded"시(디폴트) body 내용

```
title=배트맨&description=배트맨 오토바이&…
```

- binary 데이터 전송 불가

파일 업로드

❖ 폼 인코딩

- enctype="multipart/form-data"시 body 내용

```
------WebKitFormBoundaryJZ5taSn15dZXBCpl
Content-Disposition: form-data; name="title"

배트맨
------WebKitFormBoundaryJZ5taSn15dZXBCpl
Content-Disposition: form-data; name="description"

배트맨 오토바이트
------WebKitFormBoundaryJZ5taSn15dZXBCpl--

   :
```

Part

Part

❖ 파일 업로드 의존성 설정

- Commons Fileupload 1.3.1
- Commons IO 2.4

```
<!-- Apache Commons file upload -->
<dependency>
   <groupId>commons-fileupload</groupId>
   <artifactId>commons-fileupload</artifactId>
   <version>1.3.1</version>
</dependency>
<!-- Apache Commons IO -->
<dependency>
   <groupId>commons-io</groupId>
   <artifactId>commons-io</artifactId>
   <version>2.4</version>
</dependency>
```

파일 업로드

❖ MultipartResolver 설정

- MultipartResover를 스프링 설정 파일에 등록
  - Multipart로 전달된 파일 처리 지원

- servlet-context.xml

```
<beans:bean id="multipartResolver"
   class="org.springframework.web.multipart.commons.CommonsMultipartRes
olver" />
```

파일 업로드

❖ MultipartResolver 설정

- MultipartResover를 스프링 설정 파일에 등록
  - CommonsMultipartResolver의 프로퍼티

프로퍼티	타입	설명
maxUploadSize	long	최대 업로드 가능한 바이트 크기. -1은 제한이 없음을 의미. 기본값은 -1
maxInMemorySize	int	디스크에 임시 파일을 생성하기 전에 메모리에 보관할 수 있는 최대 바이트 크기. 기본 값은 1024바이트
defaultEncoding	String	요청을 파싱할 때 사용할 캐릭터 인코딩. 지정하지 않을 경우, HttpServletRequest.setCharacterEncoding 메서드로 지정한 캐릭터 셋을 사용.

파일 업로드

❖ MultipartFile 인터페이스의 주요 메서드

- 단일 파일 업로드 처리시 사용

```
@RequestMapping(value="/upload", method=RequestMethod.POST)
public String uploadSubmit(@Valid Image image,
                  BindingResult result,
                  @RequestParam("imageFiles") MultipartFile imageFile)
            throws Exception {

   if(result.hasErrors()) return "gallery/upload";
         :


   return "redirect:lightbox";
}
```

파일 업로드

❖ MultipartFile 인터페이스의 주요 메서드

메서드	설명
`String getName( )`	파라미터 이름 리턴
`String getOriginalFilename( )`	업로드 한 파일 이름 리턴
`boolean isEmpty( )`	업로드 한 파일이 존재하지 않는 경우 true 리턴
`long getSize( )`	업로드 한 파일의 크기 리턴
`byte[ ] getBytes( )` `throws IOException`	업로드한 파일 데이터의 바이트 배열리턴
`InputStream getInputStream( )` `throws IOException`	업로드 한 파일의 InputStream 리턴 InputStream의 사용 후 close( ) 호출 필요
`void transferTo(File dest)` `throws IOException`	업로드 한 파일 데이터를 지정한 파일로 저장

파일 업로드

❖ MultipartHttpServletRequest를 이용한 얼로드 파일 접근

- 다중 파일 업로드 처리시 사용

```
@RequestMapping(value="/upload", method=RequestMethod.POST)
public String uploadSubmit(@Valid Image image,
                    BindingResult result,
                   @RequestParam("imageFiles") MultipartFile imageFiles)
                    throws Exception {
   if(result.hasErrors()) return "gallery/upload";
   :


   return "redirect:lightbox";
}
```

파일 업로드

❖ MultipartHttpServletRequest를 이용한 얼로드 파일 접근

- MultipartHttpServletRequest 인터페이스 파일 관련 주요 메서드

메서드	설명
Iterator<String> getFileNames( )	업로드 된 파일들의 이름 목록을 제공하는 iterator 리턴
MultipartFile getFile(String name)	파라미터 이름이 name인 업로드 파일 정보 리턴
List<MultipartFile> getFiles(String name)	파라미터 이름이 name인 업로드 파일 정보 목록 리턴
Map<String, MultipartFile> getFileMap( )	파라미터 이름을 키로 파라미터에 해당하는 파일 정보를 값으로 하는 Map 리턴

```
List<MultipartFile> fileList = request.getFiles("imageFiles");
for(MultipartFile file : fileList) {
   // file 로 업로드 처리
}
```

파일 업로드

❖ Image 모델 수정

```
@Data
public class Image {
   private int       imageId;
   private String    title;
   private String    description;
   private String    fileName;   // 원본 이름
   private String    newName;       // 서버에서의 이름
   private String    thumbName;  // thumbnail 이름
   private long      fileSize;
   private String    mimeType;
   private Date      regDate;

}
```

파일 업로드

❖ Images 테이블 수정

```
-- new_name 필드 추가
alter table images
add (new_name varchar2(100));

-- 기존 데이터는 file_name을 new_name으로 저장
update  images set
  new_name = file_name;

commit
```

파일 업로드

❖ image-mapper.xml 수정

```
<select id="selectList" resultType="Image"
      parameterType="Pagination"><![CDATA[
   select image_id, title, description, file_name, new_name,
         thumb_name, file_size, mime_type, reg_date from (
      select row_number() over (order by image_id) as seq,
         image_id, title, description, file_name, new_name,
         thumb_name, file_size, mime_type, reg_date
      from images
   ) where seq between #{start} and #{end}
]]></select>
```

파일 업로드

❖ image-mapper.xml 수정

```
<insert id="insert" parameterType="Image"><![CDATA[
   insert into images
   (image_id, title, description, file_name, new_name, thumb_name,
      file_size, mime_type)
   values(
      images_seq.nextval, #{title}, #{description}, #{fileName},
      #{newName}, #{thumbName}, #{fileSize},  #{mimeType}
   )
]]></insert>
```

❖ 파일 업로드

```
@RequestMapping(value="/upload", method=RequestMethod.POST)
public String uploadSubmit(@Valid Image image,
                  BindingResult result,
                  MultipartHttpServletRequest request)
   throws Exception {

   if(result.hasErrors()) return "gallery/upload";

   List<MultipartFile> fileList = request.getFiles("imageFiles");
   service.upload(image, fileList);

   return "redirect:lightbox";
}
```

파일 업로드

❖ ImageService 인터페이스

```
public interface ImageService {
    :

    boolean upload(Image image, List<MultipartFile> fileList)
          throws Exception ;
}
```

파일 업로드

❖ ImageServiceImpl

```
@Override
public boolean upload(Image image, List<MultipartFile> fileList )
         throws Exception {

   for(MultipartFile file : fileList) {
      if(!file.isEmpty()) {
         saveImage(image, file);
         dao.insert(image);
      }
   }
   return true;
}
```

파일 업로드

❖ ImageServiceImpl

```
@Override
public boolean upload(Image image, List<MultipartFile> fileList )
         throws Exception {

   for(MultipartFile file : fileList) {
      if(!file.isEmpty()) {
         saveImage(image, file);
         dao.insert(image);
      }
   }
   return true;
}
```

파일 업로드

❖ ImageServiceImpl

```
@Override
public boolean upload(Image image, List<MultipartFile> fileList )
        throws Exception {

   for(MultipartFile file : fileList) {
      if(!file.isEmpty()) {
         // 업로드 파일 저장, thumbnail 생성, Image 정보 설정
         saveImage(image, file);

         // 데이터베이스 저장
         dao.insert(image);
      }
   }
   return true;
}
```

파일 업로드

❖ Tika 라이브러리

- 파일명에서 Mimetype 결정

```
<dependency>
   <groupId>org.apache.tika</groupId>
   <artifactId>tika-core</artifactId>
   <version>1.14</version>
</dependency>
```

- 사용법

```
Tika tika = new Tika();
String mimeType = tika.detect(파일명 또는 경로명);
```

파일 업로드

❖ ImageServiceImpl

```
// 파일 저장 및 Image 객체 정보 설정
public void saveImage(Image image, MultipartFile file)
                    throws Exception {

   String fileName = file.getOriginalFilename();
   String newName = saveImage(fileName, file);
   String thumbName = saveThumb(newName);

   Tika tika = new Tika();
   String mimeType = tika.detect(fileName);

   image.setFileName(fileName);
   image.setMimeType(mimeType);
   image.setFileSize(file.getSize());
   image.setNewName(newName);
   image.setThumbName(thumbName);
}
```

파일 업로드

❖ ImageServiceImpl

```
// 이름 충돌없게 timestamp 추가하여 새로운 파일명 배정 후 저장
// 새로운 파일명 리턴
public String saveImage(String fname, MultipartFile file)
                        throws Exception{
   long fileNo = System.currentTimeMillis();
   String newName = fileNo + "_" + fname;
   String path = IMAGE_DIR + "/" + newName;

   // 업로드 된 파일명 재지정
   file.transferTo(new File(path));
   return newName;
}
```

파일 업로드

❖ ImageServiceImpl

```
// Thumbnail 파일 생성 및 Thumbnail 파일명 리턴
public String saveThumb(String fname) throws Exception{
   String thumbName = "thumbnail-" + fname;

   // Thumbnail 파일 생성
   Thumbnails
      .of(new File(IMAGE_DIR + "/" + fname))
      .crop(Positions.CENTER)   // 이미지 crop
      .size(200, 200)
      .toFile(new File(THUMB_DIR + "/" + thumbName));

   return thumbName;
}
```