Spring MVC2 17 [파일 업로드]

스프링 MVC 2편 - 백엔드 웹 개발 활용 기술

김영한

2022.11.02

파일 업로드

폼 전송의 **2**가지 방식

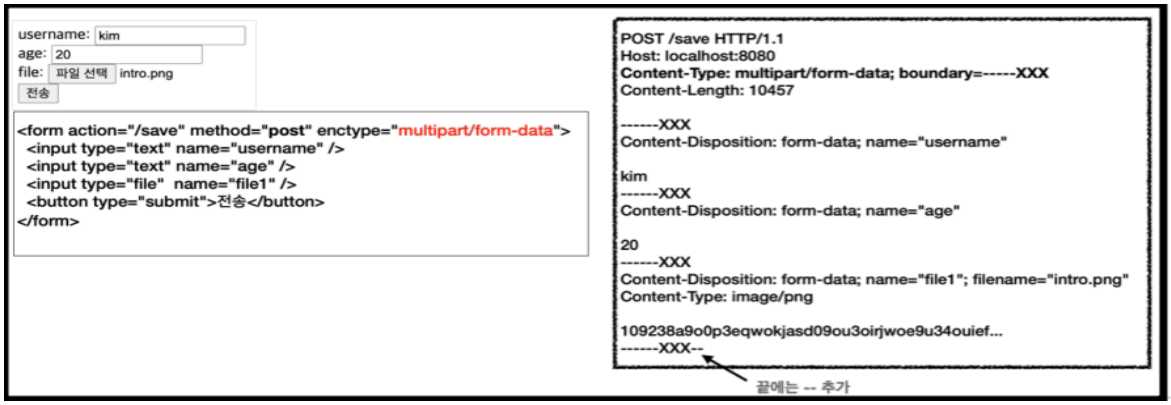
1. application/x-www-form-urlencoded

HTTP body에 문자로 & 로 구분해서 전송한다.

2. multipart/form-data

파일은 문자와 바이너리 데이터(파일) 을 같이 전송해야 한다.

이때 전송 방식을 'multipart/form-data를 사용하게 된다.



multipart : 말그대로 다른 스타일의 형식의 데이터를 전송하는 것.

```
------XXX
Content-Disposition : form-data; name = 이름  → 항목별 헤더
Content-Type

데이터 ( 바디 )
```

파일에 경우 바이너리 데이터로 전송이 된다.

→ 이러므로 서버에서 읽을 때도 복잡해 진다.

서블릿을 통한 파일 업로드

multipart/form-data

```
<div class="py-5 text-center">
   <h2>상품 등록 폼</h2>
</div>
<h4 class="mb-3">상품 입력</h4>
<form th:action method="post" enctype="multipart/form-data">
   <ul>
       <li>상품명 <input type="text" name="itemName"></li>
       <li>파일<input type="file" name="file" ></li>
   </ul>
   <input type="submit"/>
</form>
```

```
@PostMapping("/upload")
public String saveFileV1(HttpServletRequest request) throws ServletException,
IOException {
   log.info("request={}", request);

   String itemName = request.getParameter("itemName");
   log.info("itemName={}", itemName);

   Collection<Part> parts = request.getParts();
```

```
    log.info("parts={}", parts);

    return "upload-form";
}
```

파일을 upload 후 로그를 확인해 보면

```
2022-11-02 11:36:45.063  INFO 10164 --- [nio-9595-exec-1]
h.u.c.ServletUploadControllerV1          :
request=org.springframework.web.multipart.support.StandardMultipartHttpServ
letRequest@1ce4e6c2
2022-11-02 11:36:45.065  INFO 10164 --- [nio-9595-exec-1]
h.u.c.ServletUploadControllerV1          : itemName=상품A
2022-11-02 11:36:45.065  INFO 10164 --- [nio-9595-exec-1]
h.u.c.ServletUploadControllerV1          :
parts=[org.apache.catalina.core.ApplicationPart@6e3a4c9a,
org.apache.catalina.core.ApplicationPart@4b5e4d1f]
```

parts= 두개의 파트로 나눠져있는 것이 확인된다.

✔ 옵션 들

업로드 용량 설정

```
spring.servlet.multipart.max-file-size=XXMB
```

spring.servlet.multipart.enabled

```
spring.servlet.multipart.enabled=true(default)
```

multipart에 대한 처리가 복잡하기 때문에 설정에서 켜고 끌 수 있고 false로 처리하게 되면 multipart를 지원하지 않게된다.

파일을 업로드 하려면 실제 파일이 저장되는 경로가 필요

application.properties

```
file.dir=D:/Spring/file/
```

파일 경로 주입

```
@Value("${file.dir}")
private String fileDir;
```

각 파트의 정보확인

```
@PostMapping("/upload")
public String saveFileV2(HttpServletRequest request) throws ServletException,
IOException {
    log.info("request={}", request);

    String itemName = request.getParameter("itemName");
    log.info("itemName={}", itemName);

    Collection<Part> parts = request.getParts();
    log.info("parts={}", parts);

    for (Part part : parts) {
        log.info("==== PART ====");
        log.info("name={}", part.getName());
        Collection<String> headerNames = part.getHeaderNames();
        for (String headerName : headerNames) {
            log.info("header {}: {}", headerName, part.getHeader(headerName));
        }

        //편의 메서드
        //content-disposition; filename 파일 이름 꺼내는 편의 메서드를 제공
        log.info("submittedFilename={}", part.getSubmittedFileName());
        log.info("size={}", part.getSize());

        //데이터 읽기
        InputStream inputStream = part.getInputStream();
        String body = StreamUtils.copyToString(inputStream, StandardCharsets.UTF_8);
        log.info("body={}", body);
    }

    return "upload-form";
}
```

로그 확인 가능.

```
: ==== PART ====
: name=itemName
: header content-disposition: form-data; name="itemName"
: submittedFilename=null
: size=7
: body=상품A
: ==== PART ====
: name=file
: header content-disposition: form-data; name="file"; filename="02.jpg"
: header content-type: image/jpeg
: submittedFilename=02.jpg
: size=780
: body='PNG

IHDR   d      ''   sRGB '''   gAMA  ''
'a       pHYs  %  %IR$'  'IDATx^'˜''@`''''''''"+yd%' +'H$'lPHd%'='''''
'''Ig'g'i!'''''M; """""""""""
```

파일 저장

```
//파일에 저장하기
if(StringUtils.hasText(part.getSubmittedFileName())){
   String fullPath = fileDir + part.getSubmittedFileName();
   log.info("파일 저장 fullPath={}", fullPath);
   part.write(fullPath);
}
```

스프링에서 파일 업로드

```
@PostMapping("/upload")
public String saveFile(@RequestParam String itemName,
                       @RequestParam MultipartFile file, HttpServletRequest request)
throws IOException {

   log.info("request={}", request);
   log.info("itemName={}", itemName);
   log.info("multipartFile={}", file);
```

```
    if(!file.isEmpty()){
        String fullPath = fileDir + file.getOriginalFilename();
        log.info("파일 저장 fullPath={}", fullPath);
        file.transferTo(new File(fullPath));
    }

    return "upload-form";
}
```

간단하게 저장 가능.

파일 업로드, 다운로드 예제

```
<form th:action method="post" enctype="multipart/form-data">
    <ul>
        <li>상품명 <input type="text" name="itemName"></li>
        <li>첨부파일<input type="file" name="attachFile" ></li>
        <li>이미지 파일들<input type="file" multiple="multiple"
                            name="imageFiles" ></li>
    </ul>
    <input type="submit"/>
</form>
```

여러개 이미지 → multiple 옵션을 주면 List에 담길수 있다.

Item 저장할 Item class

```
@Data
public class Item {

    private Long id;
    private String itemName;
    private UploadFile attachFile;
    private List<UploadFile> imageFiles;

}
```

간단한 메모리 **Repository**

```
@Repository
public class ItemRepository {

    private final Map<Long, Item> store = new HashMap<>();
    private long sequence = 0L;

    public Item save(Item item){
        item.setId(++sequence);
        store.put(item.getId(), item);
        return item;
    }

    public Item findById(Long id){
        return store.get(id);
    }
}
```

UploadFile 클래스

```
@Data
public class UploadFile {

    //고객이 업로드 한 파일명
    private String uploadFileName;
    //서버 내부에서 관리하는 파일명
    private String storeFileName;

    public UploadFile(String uploadFileName, String storeFileName) {
        this.uploadFileName = uploadFileName;
        this.storeFileName = storeFileName;
    }
}
```

파일을 저장하는 **Component**

```
@Component
public class FileStore {

    @Value("${file.dir}")
    private String fileDir;
```

```
    public String getFullPath(String filename){
        return fileDir + filename;
    }

    public List<UploadFile> storeFiles(List<MultipartFile> multipartFiles) throws
IOException {
        List<UploadFile> storeFilesResult = new ArrayList<>();
        for (MultipartFile multipartFile : multipartFiles) {
            if(!multipartFile.isEmpty()){
                storeFilesResult.add(storeFile(multipartFile));
            }
        }
        return storeFilesResult;
    }

    public UploadFile storeFile(MultipartFile multipartFile) throws IOException {
        if(multipartFile.isEmpty()){
            return null;
        }

        String originalFilename = multipartFile.getOriginalFilename();
        String storeFileName = createStoreFileName(originalFilename);
        multipartFile.transferTo(new File(getFullPath(storeFileName)));

        return new UploadFile(originalFilename, storeFileName);
    }

    private String createStoreFileName(String originalFilename) {
        //서버 저장시 --> UUID 사용해서 생성 + 확장자를 가져오고 싶음.
        String uuid = UUID.randomUUID().toString();
        //확장자
        String ext = extractedExt(originalFilename);
        return uuid + "." + ext;
    }

    private String extractedExt(String originalFilename) {
        int pos = originalFilename.lastIndexOf(".");
        return originalFilename.substring(pos + 1);
    }
}
```

Controller설계

```
@GetMapping("/items/new")
```

```
public String newItem(@ModelAttribute ItemForm form){
   return "item-form";
}

@PostMapping("/items/new")
public String saveItem(@ModelAttribute ItemForm form, RedirectAttributes
redirectAttributes) throws IOException {

   UploadFile attachFile = fileStore.storeFile(form.getAttachFile());
   List<UploadFile> storeImageFiles = fileStore.storeFiles(form.getImageFiles());

   //데이터베이스에 저장
   Item item = new Item();
   item.setItemName(form.getItemName());
   item.setAttachFile(attachFile);
   item.setImageFiles(storeImageFiles);

   itemRepository.save(item);

   redirectAttributes.addAttribute("itemId", item.getId());

   return "redirect:/items/{itemId}";
}

@GetMapping("/items/{id}")
public String items(@PathVariable Long id, Model model){
   Item item = itemRepository.findById(id);
   model.addAttribute("item", item);
   return "item-view";
}
```

상품 조회하는 **html**

```
<div class="container">
   <div class="py-5 text-center">
       <h2>상품 조회</h2>
   </div>
   상품명: <span th:text="${item.itemName}">상품명</span><br/>
   첨부파일: <a th:if="${item.attachFile}" th:href="|/attach/${item.id}|"
th:text="${item.getAttachFile().getUploadFileName()}" /><br/>
   <img th:each="imageFile : ${item.imageFiles}"
th:src="|/images/${imageFile.getStoreFileName()}|" width="300" height="300"/>
</div> <!-- /container -->
```

저장한 이미지 표출

`th:src="|/images/${imageFile.getStoreFileName()}|"`

Controller에서 ResponseBody로 처리

```
@ResponseBody
@GetMapping("/images/{filename}")
public Resource downloadImage(@PathVariable String filename) throws
MalformedURLException {
   return new UrlResource("file:" + fileStore.getFullPath(filename));
}
```

첨부된 이미지 다운로드

```
@GetMapping("/attach/{itemId}")
public ResponseEntity<Resource> downloadAttach(@PathVariable Long itemId) throws
MalformedURLException {
   Item item = itemRepository.findById(itemId);
   String storeFileName = item.getAttachFile().getStoreFileName();
   String uploadFileName = item.getAttachFile().getUploadFileName();

   UrlResource resource = new UrlResource("file:" +
fileStore.getFullPath(storeFileName));

   log.info("uploadFileName={}", uploadFileName);

   String encodedUploadFileName = UriUtils.encode(uploadFileName,
StandardCharsets.UTF_8);
   String contentDisposition = "attachment; filename=\"" + encodedUploadFileName
+"\"";

   return ResponseEntity.ok()
           .header(HttpHeaders.CONTENT_DISPOSITION, contentDisposition)
           .body(resource);
}
```

→ 헤더를 추가해 줘야 한다.

Content-Disposition: inline (default) 로 되어 있기에 수정해 줘야 한다. 표출되는 Body의 형식을 지정