Multipart Resolver

MultipartResolver

- Dependency
  - pom.xml에 common-fileupload, common-io 라이브러리 추가

```
<!-- common fileupload -->
<dependency>
  <groupId>commons-fileupload</groupId>
  <artifactId>commons-fileupload</artifactId>
  <version>1.2.1</version>
</dependency>

<dependency>
  <groupId>commons-io</groupId>
  <artifactId>commons-io</artifactId>
  <version>1.4</version>
</dependency>
```

MultipartResolver
: 설정

- -servlet.xml에 CommonMultipartResolver를 설정

```
<!-- 멀티파트 리졸버 -->
<bean id="multipartResolver"
    class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

    <!-- 최대업로드 가능한 바이트크기 -->
    <property name="maxUploadSize" value="52428800" />

    <!-- 디스크에 임시 파일을 생성하기 전에 메모리에 보관할수있는 최대 바이트 크기 -->
    <!-- property name="maxInMemorySize" value="52428800" /-->

    <!-- defaultEncoding -->
    <property name="defaultEncoding" value="utf-8" />

</bean>
```

MultipartResolver
: 업로드 Form HTML

- 파일 업로드를 위한 HTML Form 작성
  - enctype = "multipart/form-data" 를 반드시 추가

```
<form method="post" action="" enctype="multipart/form-data">
    <label>email:</label>
    <input type="text" name="email" value="kickscar@gmail.com">
    <br><br>

    <label>파일1:</label>
    <input type="file" name="file1">
    <br><br>

    <label>파일2:</label>
    <input type="file" name="file2">
    <br><br>

    <br>
    <input type="submit" value="upload">
</form>
```

MultipartResolver
: 업로드 성공 HTML page

- 업로드 성공시 표시할 HTML Page 작성

```
<div id="content">
    <div style="margin:50px auto; width:500px;">
        <h1 style="margin-bottom:20px">Upload completed</h1>
        <div class="result-images">
            <img src="${pageContext.request.contextPath }/${urlImage }"
                style="width:150px"><br>
        </div>
        <p>
            <a href='${pageContext.request.contextPath }/fileupload/form'> 다시 업로드 하기
            </a>
        </p>
    </div>
</div>
```

MultipartResolver
: Controller에서의 처리

- File 업로드를 수행할 Controller의 작성

```
@Controller
@RequestMapping( "/fileupload" )
public class FileUploadController {
    @Autowired
    private FileUploadService fileUploadService;
    @RequestMapping( "/form" )
    public String form() {
        return "fileupload/form";
    }
    @RequestMapping( "/upload" )
    public String upload( @RequestParam String email,
        @RequestParam( "file1" ) MultipartFile file1, Model model ) {
        String saveFileName = fileUploadService.store( file1 );
        String url = "/upload-images/" + saveFileName;
        model.addAttribute( "urlImage", url );
        return "fileupload/result";
    }
}
```

MultipartResolver
: Service의 구현

- File 업로드를 수행할 Service의 작성

```
public String store(MultipartFile multipartFile) {
    String saveFilename = "";

    try {
        String originalFilename = multipartFile.getOriginalFilename();
        String extName = originalFilename.substring(originalFilename.lastIndexOf("."),
                                                    originalFilename.length());
        Long size = multipartFile.getSize();
        saveFilename = getSaveFilename(extName);      //실제 저장할 파일명 함수

        writeFile(multipartFile, saveFilename);//멀티파트 파일을 저장
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }

    return saveFilename;
}
```

MultipartResolver
: URL과 리소스의 매핑

- -servlet.xml에서 리소스 매핑 코드를 설정
  - 요청 URL의 리소스 위치를 실제 물리적 위치와 매핑

```
<!-- the mvc resources tag does the magic -->
<mvc:resources mapping="/upload-images/**" location="file:/upload/" />
```