8253jang@daum.net
장희정

파일 업로드

servlet 3.x 이전 버전에서는 아래의 라이브러리 이용.
제공하는 **MultipartRequest** 객체로 업로드기능 제공.

com.oreilly.servlet (일명 COS, cos.jar)
주소 : http://www.servlets.com/cos/

-다운로드한 라이브러리를 압축을 풀어 lib폴더에 있는
cos.jar를 파일을 이클립스 WEB-INF/lib폴더에 넣는다.

```
import com.oreilly.servlet.MultipartRequest;
import com.oreilly.servlet.multipart.DefaultFileRenamePolicy;

// 업로드 기본값 예시
String uploadDir = getServletContext().getRealPath("/upload"); // 실제 서비스에선 절대경로 권장
int maxPostSize = 10 * 1024 * 1024; // 10MB
String encoding = "UTF-8";

MultipartRequest m = new MultipartRequest(
    request,
    uploadDir,
    maxPostSize,
    encoding,
    new DefaultFileRenamePolicy() // 같은 이름이면 자동으로 (1), (2) 식으로 변경
);

String name = m.getParameter("name");
String fileSavedName = m.getFilesystemName("file"); // 서버에 저장된 파일명
java.io.File f = m.getFile("file");                 // 저장된 파일 객체
```

2) servlet 3.x 이 후 버전 부터는

자바 EE 6의 서블릿 3.0 버전부터 파일 업로드 기능을 손쉽게 해결할 수 있도록
@MultipartConfig 어노테이션과
Part인터페이스를 구현한 HttpServletRequest의 **getPart(), getParts()** 메서드를 제공한다.

8253jang@daum.net
장희정

```java
@WebServlet("/upload")
@MultipartConfig(
    location = "/path/to/tmp",          // 임시 저장 디렉터리(옵션)
    fileSizeThreshold = 1 * 1024 * 1024, // 메모리 임계치(바이트)
    maxFileSize = 10L * 1024 * 1024,     // 개별 파일 최대(바이트)
    maxRequestSize = 20L * 1024 * 1024   // 요청 전체 최대(바이트)
)
public class UploadServlet extends HttpServlet {
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, java.io.IOException {

    // 텍스트 필드는 표준적으로 getParameter 사용 가능
    String name = request.getParameter("name");
    String subject = request.getParameter("subject");

    Part part = request.getPart("file"); // <input type="file" name="file">
    String submittedFileName = part.getSubmittedFileName(); // Servlet 3.1+
```

web.xml설정

```xml
<servlet>
  <servlet-name>UploadServlet</servlet-name>
  <servlet-class>com.example.UploadServlet</servlet-class>
  <multipart-config>
    <location>/path/to/tmp</location>
    <max-file-size>10485760</max-file-size>        <!-- 10MB -->
    <max-request-size>20971520</max-request-size>  <!-- 20MB -->
    <file-size-threshold>1048576</file-size-threshold> <!-- 1MB -->
  </multipart-config>
</servlet>
```

@annotation설정

```java
@WebServlet("/upload")
@MultipartConfig(
    location = "/path/to/tmp",          // 임시 저장 디렉터리(옵션)
    fileSizeThreshold = 1 * 1024 * 1024, // 메모리 임계치(바이트)
    maxFileSize = 10L * 1024 * 1024,     // 개별 파일 최대(바이트)
    maxRequestSize = 20L * 1024 * 1024   // 요청 전체 최대(바이트)
)
public class UploadServlet extends HttpServlet {
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, java.io.IOException {
```

8253jang@daum.net
장희정

- 파일업로드 폼.

```
<form name="f" action="/upload" method="post" enctype="multipart/form-data">
  이름 : <input type="text"  name="name"><br>
  제목 : <input type="text"  name="subject"><br>
  파일첨부 : <input type="file" name="file" accept="image/*"><br><!-- 필요 시 accept -->
  <button type="submit">전송</button>
  <button type="reset">취소</button>
</form>
```

- method="post" 필수 , enctype="multipart/form-data" 필수
- `action`은 JSP가 아니라 업로드 전용 Servlet URL 로 지정 권장
- 필요하면 `accept` 속성으로 클라이언트 측 파일 선택 제한

-파일다운로드 기능(Servlet문서 작성)

1. HttpServlet 상속받는다.

2. 기능

- 업로드된 파일의 경로 알아오기
- 한글파일 인코딩 설정
- 요청된 파일이름을 File객체 변환
- ContentType설정
-실제파일을 클라이언트쪽으로 다운로드 한다.
(InputStream / OutputStream)

다운로드기능 servlet 링크 걸기

```
<a href="/contextpath/서블릿클리스이름?fileName=파일이름">
   파일이름
</a>
```

8253jang@daum.net
장희정

<다운로드 기능 코드 일부분 분석>

```
String mimeType = getServletContext().getMimeType(file.toString());

if (mimeType == null) {
    response.setContentType("application/octet-stream");
}
```

getServletContext().getMimeType(path)

- 파일의 확장자를 기준으로 **MIME 타입(콘텐츠 타입)**을 추측해서 돌려준다.
- 예:
  - "abc.pdf" → "application/pdf"
  - "cat.png" → "image/png"
  - "song.mp3" → "audio/mpeg"
- 톰캣은 기본적으로 conf/web.xml 안의 <mime-mapping> 목록을 참고해서 매칭한다.

if (mimeType == null)

- 어떤 확장자인지 모르는 경우(매핑이 없는 경우) null을 반환한다.

response.setContentType("application/octet-stream")

- MIME 타입을 지정하지 못했을 때는 기본값으로 이진 데이터(application/octet-stream)라고 브라우저에 알려준다.
- 이 타입은 "알 수 없는 바이너리"라는 뜻이라, 브라우저는 내용을 해석하려 하지 않고 다운로드용으로 처리하려는 경향이 있다.

왜 필요한가?

- 브라우저는 Content-Type을 보고 행동을 결정한다..
  - image/png → 화면에 바로 렌더링
  - application/pdf → 내장 뷰어로 열거나 다운로드
  - text/html → HTML로 해석해서 표시

- 만약 아무 MIME 타입도 지정하지 않으면?
  - 어떤 브라우저는 기본값을 추측하거나, 텍스트로 열어버릴 수도 있다.
  - 예를 들어 .exe 같은 실행파일이 잘못 열리면 보안 이슈로 이어질 수 있다.

즉, 다운로드 서블릿이라면 최소한 application/octet-stream으로 지정해 두어야
"브라우저가 파일 내용을 직접 해석하지 않고 안전하게 다운로드" 하게 된다.