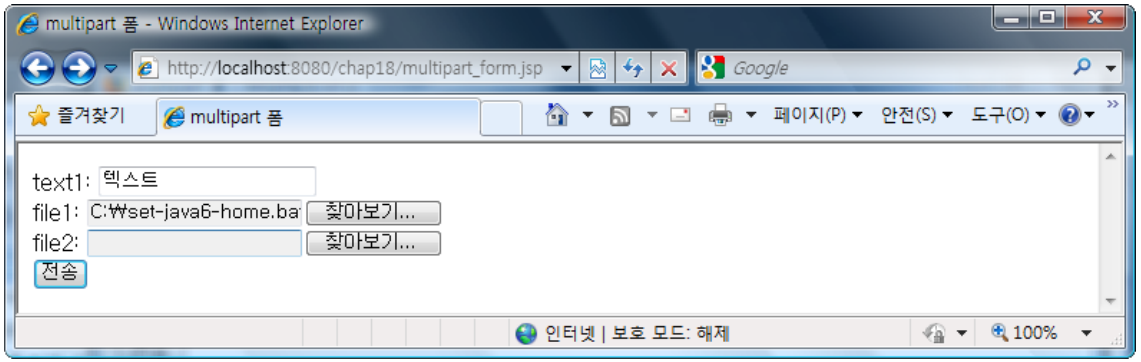
TOC

- 파일 전송 방식
- FileUpload API
- 자료실의 파일 저장 및 다운로드 구현

파일 전송을 위한 FORM 설정

- POST 방식에서 enctype을 multipart/form-data로 설정
  - type 속성이 file인 <input> 태그를 통해 파일 선택

```
<form action="..." method="post" enctype="multipart/form-data">
    ...
    <input type="file" name="file1" />
</form>
```



multipart/form-data로 전송된 데이터 형식

```
[multipart/form-data; boundary=----------------------------7d9202102082c]
------------------------------7d9202102082c
Content-Disposition: form-data; name="text1"

텍스트
------------------------------7d9202102082c
Content-Disposition: form-data; name="file1"; filename="set-java6-home.bat"
Content-Type: application/octet-stream

set JAVA_HOME=c:₩java₩jdk1.6.0_07
set PATH=%JAVA_HOME%₩bin;%PATH%
------------------------------7d9202102082c
Content-Disposition: form-data; name="file2"; filename=""
Content-Type: application/octet-stream


------------------------------7d9202102082c--
```

일반 파라미터

파일 데이터

FileUpload API

- multipart/form-data로 전송된 데이터 처리
- http://commons.apache.org/fileupload/ 에서 다운로드
  - commons-fileupload-1.2.1.jar
- 추가 필요 라이브러리
  - Commons IO
  - http://commons.apache.org/io/ 사이트에서 다운로드
  - commons-io-1.3.2.jar

FileUpload API를 이용한 업로드 데이터 처리

```
// 1. multipart/form-data 여부 확인
boolean isMultipart = ServletFileUpload.isMultipartContent(request);
if (isMultipart) {
    // 2. 메모리나 파일로 업로드 파일 보관하는 FileItem의 Factory 설정
    DiskFileItemFactory factory = new DiskFileItemFactory();
    // 3. 업로드 요청을 처리하는 ServletFileUpload 생성
    ServletFileUpload upload = new ServletFileUpload(factory);
    // 4. 업로드 요청 파싱해서 FileItem 목록 구함
    List<FileItem> items = upload.parseRequest(request);
    Iterator<FileItem> iter = items.iterator();
    while (iter.hasNext()) {
        FileItem item = iter.next();
        // 5. FileItem이 폼 입력 항목인지 여부에 따라 알맞은 처리
        if (item.isFormField()) { // 텍스트 입력인 경우
            String name = item.getFieldName();
            String value = item.getString("euc-kr");
        } else { // 파일 업로드인 경우
            String name = item.getFieldName();
            String fileName = item.getName();
            String contentType = item.getContentType();
            boolean isInMemory = item.isInMemory();
            long sizeInBytes = item.getSize();
        }
    }
}
```

업로드 한 파일 처리 방법

- FileItem.write(File file) 메서드를 사용하는 방법

- FileItem.getInputStream() 메서드로 구한 입력 스트림으로부터 바이트 데이터를 읽어와 FileOutputStream을 사용해서 파일에 출력하는 방법

- FileItem.get() 메서드로 구한 바이트 배열을 FileOutputStream을 사용해서 파일에 출력하는 방법

자료실 - 업로드 한 파일 저장 방식

- 업로드 한 파일 저장 방식 두 가지
  - DB의 BLOB에 저장
  - 파일 시스템에 저장
    - 이 경우 저장할 때 사용하는 파일의 이름은 현재 시간 등의 값을 이용해서 생성
      - 게시글 번호와 첨부된 순서 번호를 함께 사용하기도 함
    - 실제 파일 이름은 DB에 저장

자료실 - 다운로드의 구현

- 다운로드 구현 시 고려 사항
  - 응답 컨텐츠 타입은 application/octet-stream
  - Content-Disposition 헤더로 파일명 지정
    - 파일명 설정시 ISO-8859-1로 인코딩 변환해서 설정

```
<%
    String fileName = new String(item.getFileName().getBytes("euc-kr"), "iso-8859-1");
    response.setContentType("application/octet-stream");
    response.setHeader("Content-Disposition",
        "attachment; filename=₩"" + fileName+"₩"");
%>
```

- 실제 파일 전송
  - response.getOutputStream()으로 구한 OutputStream에 파일 데이터 출력