JSP 프로그래밍

(재)대덕인재개발원

p.93 필터 참고

18 파일 업로드

- 1. 파일 전송 방식
- 2. FileUpload API를 이용한 파일 업로드 구현

18.1 파일 전송 방식

- 일반 패러미터를 전송할 때 사용하는 인코딩과 파일을 업로드 할 때 사용하는 인코딩은 서로 다르다.
- HTTP의 데이터 전송방식은 크게 GET 방식과 POST 방식이 존재하는데 스트림 기반의 전송 방식인 POST 방식은 또 다시 다음의 두 가지 인코딩 방식에 따라서 전송하는 데이터 형식이 달라진다.
  - - application/x-www-form-urlencoded
  - - multipart/form-data

```
<form action=”…..” method=”post” enctype=”multipart/form-data”>
….
</form>
```

18.2 FileUpload API를 이용한 파일 업로드 구현

- FileUpload API 다운로드
  - FileUpload API는 http://commons.apache.org/fileupload/ 사이트에서 commons-fileupload-1.2.1-bin.zip 파일을 다운로드 받을 수 있다.
  - FileUploadAPI는 Commons IO API에 의존성을 가짐.
  - Commons IO는 http://commons.apache.org/io/ 사이트에서 다운 받을 수 있다.
  - 이 두 API 파일을 웹 어플리케이션의 WEB-INF/lib 디렉토리에 복사해 준다.

- FileUpload API에서 이용되는 컴포넌트.
  - FileItem : 전송된 파일객체(multipart/form-data부터 일반 폼필드 데이터까지 가지고 있음).
  - FileItemFactory : 전송된 파일이 저장될 저장소.
  - ServletFileUpload : 업로드 요청을 처리할 핸들러.

18.2 FileUpload API를 이용한 파일 업로드 구현

- FileUpload API를 이용한 multipart/form-data 처리

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page import="java.util.Iterator" %>
<%@ page import="java.util.List" %>
<%@ page import="org.apach.commons.fileupload.FileItem" %>
<%@ page import="org.apache.commons.fileupload.disk.DiskFileItemFactory" %>
<%@ page import="org.apache.commons.fileupload.servlet.ServletFileUpload" %>

<html>
<head><title>업로드 정보</title></head>
<body>
<%
// 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);
```

18.2 FileUpload API를 이용한 파일 업로드 구현

- FileUpload API를 이용한 multipart/form-data 처리

```
// 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()){
String name = item.getFieldName();
String value = item.getString("utf-8");
%>
요청 패러미터 : <%= name %>=<%= value %> <br>
<%
}else{
String name = item.getFieldName();
String filename = item.getName();
String contentType = item.getContentType();
boolean isInMemory = item.isInMemory();
long sizeInBytes = item.getSize();
%>
파일 : <%= name %>, <%= fileName %>, <%= sizeInBytes %>
<%= isInMemory ? "메모리저장" :"임시파일저장" %><br>
<%
}
```

18.2 FileUpload API를 이용한 파일 업로드 구현

- FileItem 클래스가 제공하는 메소드
  - FileItem 클래스는 multipart/form-data로 전송된 패러미터 또는 파일 정보를 저장하고 있는 클래스로서 FileItem 클래스가 제공하는 메서드를 이용해서 패러미터 이름, 값, 파일이름, 파일 데이터 등을 구할 수 있다.

메서드	리턴타입	설명
isFormField()	boolean	파일이 아닌 일반적인 입력 패러미터일 경우 true를 리턴한다.
getFieldName()	String	패러미터의 이름을 구한다.
getString()	String	기본 캐릭터 셋을 사용하여 패러미터의 값을 구한다.
getString(String encoding)	String	지정한 인코딩을 사용하여 패러미터의 값을 구한다.
getName()	String	업로드 한 파일의(경로를 제외한) 이름을 구한다.
getSize	long	업로드 한 파일의 크기를 구한다.
write(File file)	void	업로드 한 파일을 file이 나타내는 파일로 저장한다.
getInputStream()	inputStream	업로드 한 파일을 읽어오는 입력 스트림을 리턴한다.
get()	byte[]	업로드 한 파일을 byte 배열로 구한다.
isInMemory()	Boolean	업로드 한 파일이 메모리에 저장된 상태인 경우 true를 리턴하고, 임시 디렉토리에 파일로 저장된 경우 false를 리턴한다.
delete()	void	파일과 관련된 자원을 제거한다. 메모리에 저장된 경우 할당된 메모리를 반환하고 임시 파일로 저장된 경우 파일을 삭제한다.

18.2 FileUpload API를 이용한 파일 업로드 구현

- 업로드 파일을 처리하는 방법(전송된 파일을 저장하는 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
  - Commons-io API의 FileUtils이 가진 copyInputStreamToFile을 사용.

18.2 FileUpload API를 이용한 파일 업로드 구현

- 업로드 파일을 처리하는 방법

```
FileOutputStream fos = null;
InputStream is = null;
try{
        fos = new FileOutputStream(file);
        is = fileItem.getInputStream();
        byte[] buff = new byte[512];
        int len = -1;
        while((len = is.read(buff)) != -1){
                fos.write(buff, 0, len);
        }
}catch(IOException e){
        // 예외 처리
}finally{
        if(fos != null) try{ fos.close(); }catch(IOException e){}
}
```

18.2 FileUpload API를 이용한 파일 업로드 구현

- 업로드 파일을 처리하는 방법

```
FileOutputStream fos = null;
try{
        fos = new FileOutputStream(file);
        fos.write(fileItem.get());
}catch(IOException e){
        // 예외 처리
}finally{
        if(fos != null) try{ fos.close(); }catch(IOException e){}
}
```