파일 다운로더 기본 가이드

Last update: 2024/01/22

www.softbase.co.kr

목차

파일 다운로더 기본 가이드

이 화면은 FileDownload 컴포넌트에 대한 샘플 화면이다.

FileDownload 컴포넌트는 HTTP 통신 프로토콜 표준을 따르는 파일 다운로드 처리 컴포넌트이다.

CORS를 위한 with_credentials 속성 및 POST 데이터에 대한 postdata_encode 속성이 제공된다.

관련 속성으로 with_credentials, postdata_encode가 있다.

관련 API로 getfilepostdatavalue, setfilepostdatavalue, deleteallfile, clearallfilestatus, startdownload, stopdownload가 있다.

파일 정보 관련 API로 getfilecount, getfilename, getfilesize, getfilebriefsize, getfiledate, getfiletime이 있다.

파일 다운로드 상태 관련 API로 getfilestatus, getfileprogress, getfileresult, getfileresultmsg가 있다.

관련 이벤트로 on_listupdate, on_fileprogress, on_filecomplete, on_downloadcomplete, on_fileadd, on_filedelete가 있다.

예시

템플릿 위치: /HTML5/COMPONENT/FILEDOWNLOADER/filedownloader_basic

템플릿 파일

- filedownloader_basic.xml
- filedownloader_basic.js
- FileDownloader.java
- 새창으로 실행

화면 스크립트

```
// "1. 파일 추가" 버튼 이벤트
function btn_addfile_on_mouseup(objInst)
{
    var file_index;

    // 0. CORS를 위한 with_credentials 속성 및 POST 데이터에 대한 postdata_encode 속성이 제공됨

    // 1. GET 방식: HTTP URL 정보 직접 추가 방식
this.downloader_basic.addfile("http://127.0.0.1:8080/xframe5/template/HTML5/COMPONENT/FILEDOWNLOADER/150MB_1.zip", "150MB_1.zip");
this.downloader_basic.addfile("http://127.0.0.1:8080/xframe5/template/HTML5/
```

```
COMPONENT/FILEDOWNLOADER/150MB_2.zip", "150메가_2.zip");

    // 2. GET 방식: 파일 다운로드 처리를 수행하는 서블릿 호출 방식으로 URL에 GET 방식으로 쿼리 스트링 형식으로 이용하여 정보 추가
    //              xframe5 어블리케이션 설치 디렉토리 하위에 template/HTML5/COMPONENT/FILEDOWNLOADER/text.txt 파일이 있는 경우
this.downloader_basic.addfile("http://127.0.0.1:8080/xframe5/FileDownloader?DownloadFilePath=template/HTML5/COMPONENT/FILEDOWNLOADER/text.txt", "텍스트.txt");

    // 3. POST 방식: 파일 다운로드 처리를 수행하는 서블릿 호출 방식으로 POST 데이터 방식 정보 추가
    //               xframe5 어블리케이션 설치 디렉토리 하위에 template/HTML5/COMPONENT/FILEDOWNLOADER/text.txt 파일이 있는 경우
this.downloader_basic.addfile("http://127.0.0.1:8080/xframe5/FileDownloader", "텍스트_POST.txt");
    file_index = this.downloader_basic.getfileindex("텍스트_POST.txt");
    this.downloader_basic.setfilepostdatavalue(file_index, "DownloadFilePath", "template/HTML5/COMPONENT/FILEDOWNLOADER/text.txt");
}

// "모든 파일 삭제" 버튼 이벤트 처리
function btn_deleteallfile_on_mouseup(objInst)
{
    this.downloader_basic.deleteallfile();
}

// "모든 파일 상태 정보 초기화" 버튼 이벤트 처리
function btn_clearallfilestatus_on_mouseup(objInst)
{
    this.downloader_basic.clearallfilestatus();

  // 그리드에 모든 파일에 대한 상태 정보 업데이트 표시
    this.UpdateFileStauts(-1);
}

// "다운로드 시작" 버튼 이벤트 처리
function btn_startdownload_on_mouseup(objInst)
{
    this.downloader_basic.startdownload();
}

// 다운로드 중지 버튼 이벤트 처리
function btn_stopdownload_on_mouseup(objInst)
{
    this.downloader_basic.stopdownload();
}

// 파일 상태 정보 업데이터 처리
function UpdateFileStauts(nFileIndex) {
    var count, i;
```

```
    if (nFileIndex == -1) {
        count = this.downloader_basic.getfilecount();
        for (i = 0; i < count; i++) {
            this.UpdateOneFileStauts(i);
        }
    }
    else {
        this.UpdateOneFileStauts(nFileIndex);
    }
}

// 모든 다운로드 대상 파일 정보 표시
function ReloadFileStatus() {
    var nFileIndex, count;

  // 그리드의 모든 내용 삭제
    this.grdList.deleteall();

  // 파일 갯수만큼 Loop
    count = this.downloader_basic.getfilecount();
    for(nFileIndex = 0; nFileIndex < count; nFileIndex++) {
     // 행 추가
        this.grdList.additem(false, false);

     // 파일 정보 설정
        this.grdList.setitemtextex(nFileIndex, 0,
this.downloader_basic.getfilename(nFileIndex), false);
        this.grdList.setitemtextex(nFileIndex, 1,
this.downloader_basic.getfilesize(nFileIndex), false);
        this.grdList.setitemtextex(nFileIndex, 2,
this.downloader_basic.getfilebriefsize(nFileIndex), false);
        this.grdList.setitemtextex(nFileIndex, 3,
this.downloader_basic.getfiledate(nFileIndex), false);
        this.grdList.setitemtextex(nFileIndex, 4,
this.downloader_basic.getfiletime(nFileIndex), false);

     // 파일 상태 정보 설정
        this.UpdateOneFileStauts(nFileIndex, false);
    }

    this.grdList.refresh();
}

// 한 파일에 대한 다운로드 상태 표시 업데이트
function UpdateOneFileStauts(nFileIndex, bRefresh) {
    this.grdList.setitemtextex(nFileIndex, 5,
this.downloader_basic.getfilestatus(nFileIndex), bRefresh);
    this.grdList.setitemtextex(nFileIndex, 6,
this.downloader_basic.getfileprogress(nFileIndex), bRefresh);
    this.grdList.setitemtextex(nFileIndex, 7,
this.downloader_basic.getfileresult(nFileIndex), bRefresh);
    this.grdList.setitemtextex(nFileIndex, 8,
```

```
this.downloader_basic.getfileresultmsg(nFileIndex), bRefresh);
}

////////////////////////////////
// EVENT
////////////////////////////////

// 한 파일에 대한 다운로드 진행 상태 변경 이벤트 처리
function downloader_basic_on_fileprogress(objInst, nFileIndex, strFileName,
nPos)
{
    this.grdList.setitemtext(nFileIndex, 1,
this.downloader_basic.getfilesize(nFileIndex));
    this.grdList.setitemtext(nFileIndex, 2,
this.downloader_basic.getfilebriefsize(nFileIndex));
    this.grdList.setitemtext(nFileIndex, 6, nPos);
}

// 한 파일에 대한 다운로드 완료 이벤트 처리
function downloader_basic_on_filecomplete(objInst, nFileIndex, strFileName)
{
    this.grdList.setitemtext(nFileIndex, 3,
this.downloader_basic.getfiledate(nFileIndex));
    this.grdList.setitemtext(nFileIndex, 4,
this.downloader_basic.getfiletime(nFileIndex));

    this.UpdateOneFileStauts(nFileIndex);
}

function downloader_basic_on_listupdate(objInst)
{
    this.ReloadFileStatus();
}
```

FileDownloader JAVA 소스

```
package xframe5.template.html5.component.filedownloader;

import java.io.File;
import java.io.FileInputStream;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

import javax.servlet.ServletContext;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class FileDownloader extends HttpServlet {
    private static final long serialVersionUID = 3347918932345852889L;

    public void logMsg(String msg) {
        System.out.println("FileDownloader> " + msg);
    }

    public void doGet(HttpServletRequest request, HttpServletResponse
response) {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse
response) {
        FileInputStream         fis = null;
        ServletOutputStream     sos = null;

        logMsg("FileDownloader Start");

        try {
            String downloadFileAbsolutePath;

            String downloadFilePath =
request.getParameter("DownloadFilePath");

            logMsg("downloadFilePath = [" + downloadFilePath + "]");

            if (downloadFilePath.startsWith("/")) {
                downloadFileAbsolutePath = downloadFilePath;
            }
            else {
                ServletContext    context = request.getServletContext();
                String            contextAbsolutePath;

                // Root의 절대 경로를 얻어옴
                contextAbsolutePath = context.getRealPath("/");
                if(contextAbsolutePath.endsWith(File.separator) == false) {
                    contextAbsolutePath += File.separator;
                }

                downloadFileAbsolutePath = contextAbsolutePath +
downloadFilePath;
            }

            logMsg("downloadFileAbsolutePath = [" + downloadFileAbsolutePath
+ "]");

            // Set cross domain response header
```

```
            response.setHeader("Access-Control-Allow-Credentials", "true");
            response.setHeader("Access-Control-Allow-Headers", "X-Requested-
With");

            // a String containing the value of the requested header, or
null if the request does not have a header of that name
            String origin = request.getHeader("Origin");
            logMsg("origin = [" + origin + "]");

            if (origin == null) { response.setHeader("Access-Control-Allow-
Origin", "*"); }
            else { response.setHeader("Access-Control-Allow-Origin",
origin); }

            File file = new File(downloadFileAbsolutePath);
            long fileSize = file.length();

            logMsg("fileSize = [" + fileSize + "]");

            response.setHeader("Content-Length", String.valueOf(fileSize));

            long lastModified = file.lastModified();
            Date lastModifiedDate = new Date(lastModified);

            String pattern = "EEE, d MMM yyyy hh:mm:ss";
            SimpleDateFormat simpleDateFormat = new
SimpleDateFormat(pattern, new Locale("en"));

            logMsg("Last-Modified: " +
simpleDateFormat.format(lastModifiedDate));

            // response.setHeader("Last-Modified", "Tue, 21 May 2019
03:17:08 GMT");
            response.setHeader("Last-Modified",
simpleDateFormat.format(lastModifiedDate) + " GMT");

            sos = response.getOutputStream();

            byte[]     buffer = new byte[4096];
            int     numRead = -1;

            fis = new FileInputStream(new File(downloadFileAbsolutePath));
            while ((numRead = fis.read(buffer, 0, 4096)) != -1) {
                sos.write(buffer, 0, numRead);
            }

            fis.close();

            sos.flush();

            sos.close();
```

```
        }
        catch (Exception e) {
            if (fis != null) {
                try { fis.close(); }
                catch(Exception ignore) {}
            }

            if (sos != null) {
                try { sos.close(); }
                catch(Exception ignore) {}
            }

            response.setHeader("Content-Disposition", "");

            logMsg (e.getMessage());
            e.printStackTrace();
        }
    }
}
```