webfilemgr_upload 가이드

Last update: 2023/05/11

www.softbase.co.kr

목차

webfilemgr_upload 가이드

예시

템플릿 위치: /RUNTIME/COMPONENT/WEBFILEMANAGER/webfilemgr_upload

템플릿 파일

- webfilemgr_upload.xml
- webfilemgr_upload.js
- webfilemanager_upload.jsp.txt
- 새창으로 실행

화면 스크립트

```
// 툴 설치 디렉토
리\template\RUNTIME\COMPONENT\WEBFILEMANAGER\fwebfilemanager_upload.jsp.txt
-> webfilemanager_upload.jsp 이름 변경후
// 해당 파일을 WAS 서버에 적용하고, 해당 서버 URL로 아래의 strURL 변수값을 변경후 테스트 진
행해야 함
var strURL =
"http://127.0.0.1:8080/xframe5/template/RUNTIME/COMPONENT/WEBFILEMANAGER/web
filemanager_upload.jsp";
var FSO = new ActiveXObject("Scripting.FileSystemObject");

// 업로드 파일 정보 추가 버튼 이벤트 처리
function btnAddFileDirect_on_mouseup(objInst)
{
    AddUploadFileInfoToGrid("C:\\Temp\\Upload\\150MB_1.zip");
    AddUploadFileInfoToGrid("C:\\Temp\\Upload\\150MB_2.zip");
}

// 파일 선택 다이얼로그를 통한 업로드 파일 정보 추가 버튼 이벤트 처리
function btnSelectFile_on_mouseup(objInst)
{
    var strSelectFile, bMultiSelect, arrFilePath, nFilePathCount,
nFileIndex;

    bMultiSelect = true;
    strSelectFile = factory.showfileopendialog("*.*", "C:\\", "", "파일열기",
bMultiSelect);

    if (strSelectFile != "") {
        arrFilePath = strSelectFile.split("|");
        nFilePathCount = arrFilePath.length;
        for (nFileIndex = 0; nFileIndex < nFilePathCount; nFileIndex++) {
```

```
            AddUploadFileInfoToGrid(arrFilePath[nFileIndex]);
        }
    }
}

// 탐색기에서 그리드로 파일 Drop 이벤트 처리
function gridUpload_on_dropfiles(objInst, arrDropFilePath, nDropFileCount)
{
    var strFilePath, nFileIndex;

    for (nFileIndex = 0; nFileIndex < nDropFileCount; nFileIndex++) {
        AddUploadFileInfoToGrid(fso, arrDropFilePath[nFileIndex]);
    }
}

// 업로드 대상 파일 정보를 그리드에 추가
function AddUploadFileInfoToGrid(strFilePath)
{
    var objFile;

    gridUpload.insertitemtext(gridUpload.getrowcount(), 0, strFilePath);

  // 파일 오브젝트를 생성하여 파일 크기 정보를 구하여 표시
    objFile = FSO.GetFile(strFilePath);
    gridUpload.setitemtext(gridUpload.getrowcount() - 1, 1,
parseInt((objFile.Size / 1024) + 0.5, 10));
}

//업로드 시작 버튼 이벤트 처리
function btnMultiUpload_on_mouseup(objInst)
{
    var nRow, strFilePath, nUploadIndex, nResultCount;

  // 웹 파일매니저에 설정된 모든 업로드 대상 파일 목록을 삭제
    webfile.deletealluploadfile();

  // 업로드 대상 파일 정보 추가
    for (nRow = 0; nRow < gridUpload.getrowcount(); nRow++) {
        strFilePath = gridUpload.getitemtext(nRow, 0);
        if (strFilePath.length <= 0) {
            continue;
        }

        webfile.adduploadfile(strFilePath);
    }



  // 업로드 진행창 표시 여부
    var bShowProcWnd = true;

  // 업로드 시작
```

```
    if (webfile.requestupload(bShowProcWnd, strURL, "", "") == false) {
        factory.consoleprint("startupload error");
    }

  // 업로드 결과 갯수를 구함
    nResultCount = webfile.getuploadresultcount();

  // 업로드 결과 갯수만큼 Loop
    for (nUploadIndex = 0; nUploadIndex < nResultCount; nUploadIndex++) {
     // 업로드 결과 코드를 구함
        nResultCode = webfile.getuploadresultcode(nUploadIndex);

     // 업로드 결과 코드에 따라서 그리드에 결과 표시
        if (nResultCode == 1) {
            gridUpload.setitemtext(nUploadIndex, 2, "성공");
            gridUpload.setitemtext(nUploadIndex, 3,
webfile.getuploadresultfilename(nUploadIndex));
        }
        else if (nResultCode == 0) {
            gridUpload.setitemtext(nUploadIndex, 2, "오류메세지 수신");
            gridUpload.setitemtext(nUploadIndex, 3,
webfile.getuploadresulterrormsg(nUploadIndex));
        }
        else {
            gridUpload.setitemtext(nUploadIndex, 2, "오류");
            gridUpload.setitemtext(nUploadIndex, 3,
webfile.getuploadresulterrormsg(nUploadIndex));
        }
    }

  // 업로드 후 업로드 결과에 대한 정보 및 수신 데이터를 담고 있는 정보들을 모두 지우는 API이다.
  // 업로드의 모든 처리가 끝난 후에는 해당 정보를 삭제 하는게 메모리 관리에 유리하다.
    webfile.deletealluploadresultinfo();

    btnMultiUpload.setfocus();
}
```

webfilemanager_upload.jsp 스크립트

```
<%@ page import="java.io.File" %>
                <%@ page import="java.io.IOException" %>
                <%@ page import="java.io.PrintWriter" %>
                <%@ page import="java.io.UnsupportedEncodingException" %>
                <%@ page import="java.util.HashMap" %>
                <%@ page import="java.util.Iterator" %>
                <%@ page import="java.util.List" %>

                <%@ page import="org.apache.commons.fileupload.FileItem" %>
                <%@ page
```

```
import="org.apache.commons.fileupload.FileUploadException" %>
                <%@ page
import="org.apache.commons.fileupload.disk.DiskFileItemFactory" %>
                <%@ page
import="org.apache.commons.fileupload.servlet.ServletFileUpload" %>
                <%@ page import="org.apache.commons.io.FilenameUtils" %>
                <%@ page import="org.apache.log4j.Logger" %>

                <%!
                    // create logging object
                    Logger     logger = Logger.getLogger(getClass());

                    // Define WebFileManager(WFM) Constant
                    String    WFM_DATA_DEL        =
String.valueOf((char)0x1A);        // data delimeter
                    String    WFM_DATASTART_DEL    =
String.valueOf((char)0x1C);        // data start indicator예
                    String    WFM_DATAEND_DEL        =
String.valueOf((char)0x1F);        // data end indicator

                    int        maxMemoryFileSize = 10;                //
maximum memory file size
                    int        maxFileSize =  1000 * 1024 * 1024;    //
maximum file size (100MB)

                    String    errorMsg = "";
                    String    contextRootDir = "";
                    String    tempDirAbsolutePath = "";
                    String    saveBaseDirAbsolutePath = "";
                %>

                <%
                    ServletFileUpload     uplaodHandler = null;
// file uplaod handler
                    List                fileItemList = null;            //
file item list
                    FileItem             fileItem = null;               //
file item
                    HashMap                paramMap = new HashMap();
// parameter map

                    String                 strResultMsg = WFM_DATASTART_DEL;

                    out.clearBuffer();

logger.info("============================================================"
);
                    logger.info("webfilemanager upload start");

                    // Set cross domain response header
                    response.setHeader("Access-Control-Allow-Credentials",
```

```
"true");
                    if (request == null) {
                        logger.error("Access-Control-Allow-Origin = *");
                        response.setHeader("Access-Control-Allow-Origin",
"*");
                    }
                    else {
                        logger.error("Access-Control-Allow-Origin = " +
request.getHeader("Origin"));
                        response.setHeader("Access-Control-Allow-Origin",
request.getHeader("Origin"));
                    }
                    response.setHeader("Access-Control-Allow-Headers", "X-
Requested-With");

                    // Set upload file temp dir, save base dir path
                    // setUploadEnvSetting(getServletContext());        //
servlet 3.0 spec
setUploadEnvSetting(request.getSession().getServletContext());

                    // Check whether the request has multipart data content.
                    boolean isMultipart =
ServletFileUpload.isMultipartContent(request);
                    if (!isMultipart) {
                        try {
                            logger.error("There is no multipart data in
request");

                            strResultMsg += getErrorMsg("", "There is no
multipart data in request");
                            strResultMsg += WFM_DATAEND_DEL;

                            out.print(strResultMsg);
                        }
                        catch (UnsupportedEncodingException e) {
                            e.printStackTrace();
                        }
                        return;
                    }

                    // Create a file upload handler
                    uplaodHandler = getFileUploadProcessor();
                    logger.info("Success to create a file upload handler");

                    // parse the request
                    try {
                        fileItemList = uplaodHandler.parseRequest(request);
                    }
                    catch (FileUploadException ex) {
                        try {
                            logger.error("Fail to parse a request");
```

```
                            strResultMsg += getErrorMsg("", "Fail to parse a
request");
                            strResultMsg += WFM_DATAEND_DEL;

                            out.print(strResultMsg);
                        }
                        catch (UnsupportedEncodingException e) {
                            logger.error("Exception msg = " +
e.getMessage());
                            e.printStackTrace();
                        }
                        return;
                    }

                    logger.info("Success to parse a request");

                    // Process the parameter
                    Iterator iter1 = fileItemList.iterator();
                    while (iter1.hasNext()) {
                        fileItem = (FileItem)iter1.next();

                        // process a regular form field
                        if (fileItem.isFormField()) {
                            String fieldName = fileItem.getFieldName();
                            String fieldValue = fileItem.getString("UTF-8");

                            logger.info("fieldName = " + fieldName + ",
fieldValue = " + fieldValue);
                            paramMap.put(fieldName, fieldValue);
                        }
                    }

                    Iterator iter2 = fileItemList.iterator();
                    while (iter2.hasNext()) {
                        fileItem = (FileItem)iter2.next();

                        // process a regular form field
                        if (fileItem.isFormField()) {
                            continue;
                        }

                        String fileItemFileName = fileItem.getName();
                        String fileResultMsg = null;

                           try {
                               // process a upload multipart data
                               String saveFileName =
handleUploadFile(fileItem, paramMap);
                               if (saveFileName == null) {
                                   fileResultMsg =
getErrorMsg(fileItemFileName, errorMsg);
```

```
                                }
                                else {
                                 // make a success message
                                    fileResultMsg =
getSuccessMsg(fileItemFileName, saveFileName);
                                }

                                logger.info("fileResultMsg = [" + fileResultMsg
+ "]");

                                // add one file process message to return
message
                                    strResultMsg += fileResultMsg;
                               }
                               catch (Exception ex) {
                                logger.error("Exception Msg = " +
ex.getMessage());
                                ex.printStackTrace();
                                fileResultMsg = getErrorMsg(fileItemFileName,
"Fail to process upload file.");

                                // add one file process message to return
message
                                    strResultMsg += fileResultMsg;
                               }
                        }

                        // add data end delimiter
                        strResultMsg += WFM_DATAEND_DEL;

                        logger.info("strResultMsg = [" + strResultMsg + "]");

                        out.print(strResultMsg);
                    %>

                    <%!
                    // Set upload file temp dir, save base dir path
                    public void setUploadEnvSetting(ServletContext context)
                    {
                        // get context real path
                        contextRootDir = context.getRealPath("/");
                        if(contextRootDir.endsWith(File.separator) == false) {
                            contextRootDir += File.separator;
                        }

                        // temporary directory absolute path for temporary file
                        tempDirAbsolutePath = contextRootDir + "temp";
                        saveBaseDirAbsolutePath = contextRootDir + "upload";

                        logger.info("tempDirAbsolutePath = " +
tempDirAbsolutePath);
```

```
                    logger.info("saveBaseDirAbsolutePath = " +
saveBaseDirAbsolutePath);

                    makeDirUsingDirPath(tempDirAbsolutePath);
                    makeDirUsingDirPath(saveBaseDirAbsolutePath);

                    return;
                }

                // handle a upload file data
                public String handleUploadFile(FileItem fileItem, HashMap
paramMap) throws Exception
                {
                    String        saveFileAbsolutePath = "";        // file
absolute path to save
                    String        saveFileName = "";

                    String         filePath = fileItem.getName();
// HTML??File Item Name
                    String        fileName =
FilenameUtils.getName(filePath);    // file name

                    String         paramDirPath = "";
                    String         paramFileName = "";

                    logger.info("filePath = " + filePath);
                    logger.info("fileName = " + fileName);
                    logger.info("getContentType = " +
fileItem.getContentType());
                    logger.info("getSize = " + fileItem.getSize());

                    // get file save information
                    paramDirPath = paramMap.get("DIR_PATH") == null ? "" :
(String)paramMap.get("DIR_PATH");
                    paramFileName = paramMap.get("FILE_NAME") == null ? "" :
(String)paramMap.get("FILE_NAME");

                    logger.info("paramDirPath = [" + paramDirPath + "]");
                    logger.info("paramFileName = [" + paramFileName + "]");

                    // set save directory absolute path
                    saveFileAbsolutePath = saveBaseDirAbsolutePath;
                    if (paramDirPath.length() > 0) {
                        saveFileAbsolutePath = saveFileAbsolutePath +
File.separatorChar + paramDirPath;
                    }

                    // set save file absolute path
                    if (paramFileName.length() > 0) {
                        saveFileName = paramFileName;
                    }
```

```
                    else {
                        saveFileName = fileName;
                    }
                    saveFileAbsolutePath = saveFileAbsolutePath +
File.separator + saveFileName;

                    logger.info("saveFileAbsolutePath = " +
saveFileAbsolutePath);

                    // make a directory for file path
                    makeDirUsingFilePath(saveFileAbsolutePath);

                    // save a upload file to a save file absolute path
                    while (true) {
                        int    retryCount = 0;

                        try {
                            fileItem.write(new File(saveFileAbsolutePath));
                            break;
                        }
                        catch(Exception e) {
                            logger.error("Exception Msg = " +
e.getMessage());
                            retryCount++;

                            if (retryCount > 5) {
                                logger.error("Fail to wirte a file");
                                errorMsg = "Fail to wirte a file";
                                return null;
                            }
                            else {
                                try {
                                    Thread.sleep(1000);
                                }
                                catch (InterruptedException ignore) {
                                    ;
                                }
                                continue;
                            }
                        }
                    }

                    logger.info("Success To Wirte File, saveFileName = " +
saveFileName);

                    // delete a file item content?
                    fileItem.delete();

                    // return a saved file name
                    return saveFileName;
                }
```

```
                //TODO: change a character set of message
                public String msgCharacterSetConvert(String message) throws
UnsupportedEncodingException
                {
                    if (message == null) {
                        return "";
                    }

                    /*
                    logger.info("don't change character set");
                    return message;
                    */

                    logger.info("change character set UTF-8 -> ISO-8859-1");
                    return new String(message.getBytes("UTF-8"),
"ISO-8859-1");
                }

                // make a success message
                public String getSuccessMsg(String fileName, String
saveFileName) throws UnsupportedEncodingException {
                    StringBuffer    returnMsg = new StringBuffer();

                    returnMsg.append("Result=\"success\"&");
                    returnMsg.append("FileName=\"" + fileName + "\"&");
                    returnMsg.append("SaveFileName=\"" + saveFileName +
"\"");


                    returnMsg.append(WFM_DATA_DEL);

                    return msgCharacterSetConvert(returnMsg.toString());
                }

                // make a error message
                public String getErrorMsg(String fileName, String errorMsg)
throws UnsupportedEncodingException
                {
                    StringBuffer    returnMsg = new StringBuffer();

                    returnMsg.append("Result=\"error\"&");
                    returnMsg.append("FileName=\"" + fileName + "\"&");
                    returnMsg.append("Message=\"" + errorMsg + "\"");

                    returnMsg.append(WFM_DATA_DEL);

                    return msgCharacterSetConvert(returnMsg.toString());
                }

                private String getRandomFileName() {
                    return java.util.UUID.randomUUID().toString().replace("-
", "");
```

```
                }

                // get a file upload process
                private ServletFileUpload getFileUploadProcessor()
                {
                    // create a new file item factory
                    DiskFileItemFactory factory = new DiskFileItemFactory();

                    // create a new file object for a temporary directory
                    File    tempDir = new File(tempDirAbsolutePath);

                    // create a temporary directory
                    if(!tempDir.exists()) {
                        tempDir.mkdirs();
                    }

                    // set a temporary directory
                    factory.setRepository(tempDir);

                    // set a maximum memory file size
                    factory.setSizeThreshold(maxMemoryFileSize);

                    // create a servlet file upload object using a factory
                    ServletFileUpload upload = new
ServletFileUpload(factory);

                    // set a maixmum file size
                    upload.setSizeMax(maxFileSize);

                    // set header encoding characterset for hangul file name
                    upload.setHeaderEncoding("UTF-8");

                    return upload;
                }

                // make a directory using a file path
                public void makeDirUsingFilePath(String fileAbsolutePath)
                {
                    File     oFile = new File(fileAbsolutePath);
                    File    oDir = oFile.getParentFile();

                    // create a diectory
                    if (!oDir.exists()) {
                        oDir.mkdirs();
                    }
                }

                // make a directory using a dir path
                public void makeDirUsingDirPath(String dirAbsolutePath)
                {
                    File     oDir = new File(dirAbsolutePath);
```

```
                    // create a diectory
                    if (!oDir.exists()) {
                        oDir.mkdirs();
                    }
                }
                %>
```