엑셀 처리 가이드

Last update: 2024/04/24

목차

엑셀 처리 가이드

엑셀 연동 방식 개요

엑셀 연동 방식에 대한 개요 및 구조는 아래와 같습니다. 연동 방식에 대한 변경은 실행 파라미터에 의해서 결정됩니다.

엑셀 연동 방식 개요

구분	WAS 서버 연동 방식	로컬 엑셀 오토메이션 방식	로컬 스크립트 방식
기반 기술	• xFrame5 EXCEL 처리 스크립트 모듈 • Apache POI 라이브러리 • xFrame5 XEXCEL 자바 라이브러리	• xFrame5 EXCEL 처리 스크립트 모듈 • 엑셀 오토메이션 기술 • xFrame5 전용 브라우저	• xFrame5 EXCEL 처리 스크립트 모듈 • 엑셀 처리용 오픈소스 자바스크립트 라이브러리
xFrame5 실행 환경	• 범용 브라우저, xFrame5 전용 브라우저	• xFrame5 전용 브라우저 (범용 브라우저 불가)	• 범용 브라우저, xFrame5 전용 브라우저
실행 환경	• Java 기반 WAS 서버 • JDK 1.6 이상 • 고객사에 적용된 DRM 솔루션 회사에서 제공하는 DRM 해제/설정 자바 라이브러리	• 사용자 PC에 엑셀 프로그램이 설치되어 있어야 함	• 실행 환경제약 사항 없음 • 기본 방식으로는 DRM 해제/설정 불가
처리 속도	• 보통	• 조금 우수 (엑셀 오토메이션 작업으로 인한 처리 시간 증대)	• 우수
서버 부하	• 대량 데이터 및 사용자가 많은 경우, 서버에 부하(메모리, CPU) 발생	• 서버 부하 없음	• 서버 부하 없음
파일 선택 및 저장 경로	• 사용자 선택 (직접 경로 지정 불가)	• 사용자 선택 및 직접 경로 지정 가능	• 사용자 선택 (직접 경로 지정 불가)

DRM 처리

DRM 처리 방식 상세 내용

엑셀 연동 방식에 대한 개요 및 구조는 아래와 같습니다. 연동 방식에 대한 변경은 실행 파라미터에 의해서 결정됩니다.

구분	WAS 서버 연동 방식	로컬 엑셀 오토메이션 방식	로컬 스크립트 방식
엑셀 파일 업로드 DRM 해제	• DRM 솔루션 회사에서 제공하는 DRM 해제 자바 라이브러리를 연동하여 xFrame5 XEXCEL 자바 라이브러리에서 DRM 해제 수행	• 엑셀 프로그램에서 DRM 해제	기본적으로 DRM 해제 불가로 인해 엑셀 업로드 동작 오류 발생 [DRM 해제 방안] - 방안1 (권고 방안) xFrame5 전용 브라우저인 경우, DRM 솔루션에서 xFrame5 전용 브라우저를 엑셀 프로그램처럼 DRM 자동 해제 방식 적용 가능 - 방안 2 xFrame5 전용 브라우저인 경우, 엑셀 업로드시 DRM 해제를 위한 DRM 솔루션 연계 커스트마이징 개발 필요 - 방안 3 GRID_EXCELUPLOAD_METHOD2_DRMURL 실행 파라미터를 지정하여, 엑셀 업로드시 DRM 솔루션 연계한 DRM 해제 처리 모듈(하단 XDrmUtil.java 소스 참조) 커스트마이징 개발
엑셀 파일 다운로드 DRM 적용	• DRM 솔루션 회사에서 제공하는 DRM 적용 자바 라이브러리를 연동하여 xFrame5 XEXCEL 자바 라이브러리에서 수행	• 엑셀 저장시 DRM 적용이 필요한 경우, “로컬 스크립트 방식”으로 엑셀 파일 저장 후 • 엑셀 프로그램 기동 및 저장 작업을 수행하여 DRM 적용	기본적으로 엑셀 파일 저장시 DRM 적용 안됨 [DRM 적용 방안] - 방안 1 (권고 방안) DRM 솔루션에서 자동으로 DRM 적용 처리 - 방안 2 xFrame5 전용 브라우저인 경우, 엑셀 저장시 DRM 적용을 위한 DRM 솔루션 연계 커스트마이징 개발 필요 - 방안 3 GRID_EXCELDOWNLOAD_METHOD2_DRMURL 실행 파라미터를 지정하여, 엑셀 저장시 DRM 솔루션 연계한 DRM 적용 처리 모듈(하단 XDrmUtil.java 소스 참조) 커스트마이징 개발
고객사 보안 솔루션 정책 확인	• 범용 브라우저/xFrame5 전용 브라우저에서 WAS 서버로 파일 업로드 보안 허용 • 범용/xFrame5 전용 브라우저에서 파일 다운로드 보안 허용	xFrame5 전용 브라우저에서 파일 다운로드 보안 허용	범용 브라우저/xFrame5 전용 브라우저에서 파일 다운로드 보안 허용

엑셀 업로드 상세 방식

엑셀 업로드 상세 방식

엑셀 업로드 방식에 대한 자세한 내용은 아래와 같습니다.

구분	WAS 서버 연동 방식	로컬 엑셀 오토메이션 방식	로컬 스크립트 방식
엑셀 업로드 절차	• 사용자가 엑셀 파일 선택 (API에서 직접 경로 지정 불가) • 사용자가 선택한 파일을 WAS 서버에 HTTP 멀티 파트 형태로 업로드 • WAS 서버의 xFrame5 XEXCEL 자바 라이브러리에서 업로드된 엑셀 파일 파싱 및 데이터 추출하여 리턴 • GRID에서 수신된 데이터 처리 및 표시	• 사용자가 엑셀 파일 선택 (API에서 직접 경로 지정 가능) • xFrame5 전용 브라우저에서 엑셀 프로그램을 기동하여 엑셀 프로그램에서 엑셀 파일 로드 후, 엑셀 프로그램과의 인터페이스를 통해 데이터 추출 및 리턴 • GRID에서 추출된 데이터 처리 및 표시	• 사용자가 엑셀 파일 선택 (API에서 직접 경로 지정 불가) • xFrame5 XEXCEL 스크립트 라이브러리에서 사용자가 선택한 엑셀 파일 파싱 데이터 추출 및 리턴 • GRID에서 추출된 데이터 처리 및 표시
실행 파라미터	GRID_EXCELUPLOAD_METHODTYPE 파라미터 • 엑셀 다운로드 방식 지정 • 파라미터 값 : 0 (기본값)	GRID_EXCELUPLOAD_METHODTYPE 파라미터 • 엑셀 업로드 방식 지정 • 파라미터 값 : 1	GRID_EXCELUPLOAD_METHODTYPE 파라미터 • 엑셀 업로드 방식 지정 • 파라미터 값 : 2

엑셀 다운로드 상세 방식

엑셀 다운로드 상세 방식

엑셀 다운로드 방식에 대한 자세한 내용은 아래와 같습니다.

구분	WAS 서버 연동 방식	로컬 엑셀 오토메이션 방식	로컬 스크립트 방식
엑셀 다운로드 절차	• 그리드의 내용을 WAS 서버에 HTTP POST 형태로 송신 • WAS 서버의 xFrame5 XEXCEL 자바 라이브러리에서 수신된 데이터를 이용하여 엑셀 파일 생성 및 엑셀 파일 리턴 • 수신된 엑셀 파일 저장 • 경로 지정 파일 저장 불가 (저장시 사용자가 폴더 지정 경로 지정)	• 그리드의 내용을 xFrame5 XEXCEL 스크립트 라이브러리에서 엑셀 데이터 생성하여 xFrame5 전용 브라우저에게 전달 • xFrame5 전용 브라우저에서 엑셀 프로그램 기동 및 저장 • 경로 지정 파일 저장 가능	• XLSX 파일 형식만 지원 • 그리드의 내용을 xFrame5 XEXCEL 스크립트 라이브러리에서 엑셀 파일 생성 및 엑셀 파일 저장 • 경로 지정 파일 저장 불가 (파일 저장시 사용자가 폴더 지정 경로 지정)
실행 파라미터	GRID_EXCELDOWNLOAD_METHODTYPE 파라미터 • 엑셀 다운로드 방식 지정 • 파라미터 값 : 0 (기본값)	GRID_EXCELDOWNLOAD_METHODTYPE 파라미터 • 엑셀 다운로드 방식 지정 • 파라미터 값 : 1	GRID_EXCELDOWNLOAD_METHODTYPE 파라미터 • 엑셀 다운로드 방식 지정 • 파라미터 값 : 2
세부 옵션 실행 파라미터	XEXCEL_DOWNLOAD_URL 파라미터 • XExcelDownload 서블릿 URL 지정 • 예시 - "./xframe5/XExcelDownload"		
엑셀 파일 비밀번호 설정	• XExcelUpload 서블릿에서 엑셀 파일 비밀번호 지정 처리		• 로컬 스크립트에서 엑셀 파일 비밀번호 지정 처리 • 처리 시간이 조금 더 걸림

제약 사항

상세 내용

엑셀 연동 관련 제약 사항에 대한 자세한 내용은 아래와 같습니다.

구분	WAS 서버 연동 방식	로컬 엑셀 오토메이션 방식	로컬 스크립트 방식
엑셀 업로드 파일크기	• 그리드 excel_upload_maxsize 속성 • 그리드 get/setuploadexcelmaxsize API 제공	• N/A	• N/A
실행 파라미터	GRID_EXCELUPLOAD_MAXSIZE 파라미터 • 업로드할 파일 최대 크기 지정 • 기본값: 0 (제한없음) • 단위: 바이트 • 그리드 excel_upload_maxsize 속성이 0인 경우 사용됨		

XDrmUtil.java 소스

XDrmUtil.java 파일은 GRID_EXCELUPLOAD_METHODTYPE/GRID_EXCELDOWNLOAD_METHODTYPE 실행 파라미터가 2인 경우, 각각 엑셀 파일에 대한 DRM 해제 및 적용 처리를 위한 샘플 자바 서블릿 소스이다.

프로젝트 진행시에는 고객사의 DRM 솔루션과의 연계 로직을 추가하여 사용해야 한다.

소스 파일

- XDrmUtil.java

소스

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

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLEncoder;

import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.InputStream;
import java.io.PushbackInputStream;
import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

import javax.servlet.ServletContext;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.output.DeferredFileOutputStream;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

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

    /*****************************************************************
   * 프로젝트 적용시 변경 대상 시작
     *****************************************************************/
    protected final Log logger = LogFactory.getLog(getClass());

   // 업로드된 파일 처리용 임시 파일을 저장할 디렉토리 절대 경로 (프로젝트 적용시 변경필요)
    String    tempDirAbsolutePath = "C:\\xFrame5\\temp";

   // 업로드된 파일을 임시로 저장할 디렉토리 절대 경로 (프로젝트 적용시 변경필요)
    String    saveBaseDirAbsolutePath = "C:\\xFrame5\\temp";

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

    /**
   * 메시지 로깅
     * @return
     */
    public void logMsg(String msg)
    {
        // logger.info(msg);
        System.out.println(msg);
    }

    /**
   * 파일 업로드 처리시 사요한 임시 저장 디렉토리 및 파일 저장 디렉토리 설정
     * (getContextRootPath 함수 활용)
     * @return
     */
    public void setTempAndSaveDirPath(HttpServletRequest request)
    {
        String contextRootAbsolutePath;

        // Web Application이 실행되는 루트 절대 경로를 구함
        contextRootAbsolutePath = getContextRootPath(request);

        if (contextRootAbsolutePath.endsWith(File.separator) == false) {
            contextRootAbsolutePath += File.separator;
        }

        logMsg("contextRootPath = [" + contextRootAbsolutePath + "]");

     // 업로드된 파일 처리용 임시 파일을 저장할 디렉토리 절대 경로 (프로젝트 적용시 변경필요)
        // tempDirAbsolutePath = "C:\\xFrame5\\temp";
        tempDirAbsolutePath = contextRootAbsolutePath + "temp";
        logMsg("tempDirAbsolutePath = [" + tempDirAbsolutePath + "]");

     // 업로드된 파일을 임시로 저장할 디렉토리 절대 경로 (프로젝트 적용시 변경필요)
        // saveBaseDirAbsolutePath = "C:\\xFrame5\\temp";
        saveBaseDirAbsolutePath = contextRootAbsolutePath + "temp";
        logMsg("saveBaseDirAbsolutePath = [" + saveBaseDirAbsolutePath +
"]");
    }

    /**
     * Request에서 전달된 파라미터 KEY/VALUE 맵과 저장된 파일 경로를 이용하여
     * DRM 적용/해제 및 DRM 적용/해제된 파일 절대 경로 반환
     * @param paramMap Request에서 전달된 파라미터 KEY/VALUE 맵
     * @param saveFileAbsolutePath 저장된 파일 경로
     * @return  DRM 적용/해제된 파일 절대 경로 반환
     */
    public String drmProcess(HashMap<String, String> paramMap, String
saveFileAbsolutePath)
    {
        String returnFileAbsolutePath;

        // TODO: DRM 적용 또는 해제 처리 및 처리 결과 파일 경로 반환
        returnFileAbsolutePath = saveFileAbsolutePath;

        return returnFileAbsolutePath;
    }

    /***************************************************************
   * 프로젝트 적용시 변경 대상 끝
     ***************************************************************/
    String    errorMsg = "";        // error message
    String    fileName = "";        // uplaod file name
    String  browserType = ""; // browser type

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

    public void doPost(HttpServletRequest request, HttpServletResponse
response) {
        ServletFileUpload     uplaodHandler = null;            // file
uplaod handler
        List<?>                fileItemList = null;            // file item
list
        FileItem              fileItem = null;                // file item
        String                saveFileAbsolutePath = "";

        byte[]                 buffer = null;
        int                   numRead = -1;

        FileInputStream         fis = null;
        ServletOutputStream     sos = null;

        // POST 방식으로 전달되는 파일 이외의 값을 저장하기 위한 HashMap
        HashMap<String, String> paramMap = new HashMap<String, String>();

     // 임시로 파일 이름 및 절대 경로를 지정함
     // 실제 환경에서는 request에서 파일 이름, 저장할 파일 이름을 구하고, 파일 이름에 해당하는
절대 경로를 구해야 함.
        String                 returnFileAbsolutePath = "";

        logMsg("XDrmUtil Start ---------------------------------------------
-------");

        try {
            // Get Response Output Stream
            sos = response.getOutputStream();

            setTempAndSaveDirPath(request);

            // Set Response Common Header
            setResponseCommonHeader(request, response);

            // Get Upload File Processor Object
            uplaodHandler = getFileUploadProcessor();
            logMsg("Success to create a file upload handler");

            // Parse request Data
            fileItemList = uplaodHandler.parseRequest(request);

            // Get Parameter Key/Value Map
            getParamValueMap(fileItemList, paramMap);

            // Process File Data
            Iterator<?> iter = fileItemList.iterator();
            while (iter.hasNext()) {
                fileItem = (FileItem)iter.next();
                // Skip Normal Form Field Item
                if (fileItem.isFormField()) { continue; }

                   // Save Upload File Multipart Data
                   saveFileAbsolutePath = handleUploadFile(fileItem,
paramMap);
                   if (saveFileAbsolutePath == null) {
                       logMsg("saveFileAbsolutePath is null, errorMsg = " +
errorMsg);
                       returnErrorMsg(response, sos, errorMsg);
                    return;
                   }

                   break;
            }

            // DRM 적용/해제 처리 및 처리 결과 파일 경로를 returnFileAbsolutePath 변
수에 저장
            returnFileAbsolutePath = drmProcess(paramMap,
saveFileAbsolutePath);

            // Create Byte Array For Read File
            buffer = new byte[4096];

            // Create File Object For Return File Path
            File file = new File(returnFileAbsolutePath);
            long fileSize = file.length();

            // Set HTTP Response HTTP Header
            setResponseHeader(response, fileName, fileSize);

            // Read Return File Content & Write File Content to Output
Stream
            fis = new FileInputStream(new File(returnFileAbsolutePath));
            while((numRead = fis.read(buffer, 0, 4096)) != -1){
                sos.write(buffer, 0, numRead);
            }

            sos.flush();
        }
        catch (Exception e) {
            logMsg(e.getMessage());
            e.printStackTrace();

            // Return Error Message
            try { returnErrorMsg(response, sos, e.getMessage()); }
            catch (Exception ignore) {}
        }
        catch (Throwable e) {
            logMsg(e.getMessage());
            e.printStackTrace();
        }
        finally {
            // Close File Input Stream
            if (fis != null) {
                try { fis.close(); }
                catch (Exception ignore) {}
            }

            // Close Servlet Output Stream
            if (sos != null) {
                try { sos.close(); }
                catch (Exception ignore) {}
            }

            logMsg("delete " + saveFileAbsolutePath);
            deleteFile(saveFileAbsolutePath);
            logMsg("delete " + returnFileAbsolutePath);
            deleteFile(returnFileAbsolutePath);
        }
    }

    public void returnErrorMsg(HttpServletResponse response,
ServletOutputStream sos, String errorMsg) throws IOException
    {
    // 에러 발생시 메세지 리턴용
        //     StartDelimiter + 에러메시지 + EndDelimiter
        String    WFM_DATASTART_DEL    = String.valueOf((char)0x1C) +
"xFrame5ErrorS" + String.valueOf((char)0x1C);    // data start indicator
        String    WFM_DATAEND_DEL        = String.valueOf((char)0x1F) +
"xFrame5ErrorE" + String.valueOf((char)0x1F);    // data end indicator

        String errMsg = WFM_DATASTART_DEL + errorMsg + WFM_DATAEND_DEL;

        byte[] arrReturn = errMsg.getBytes("UTF-8");

        response.setContentType("text/html");
        response.setHeader("Content-Length",
String.valueOf(arrReturn.length));

        sos.write(arrReturn);
    }

    // request 파싱 처리 결과 리스트에서 파일 데이터를 제외한 일반 KEY/VALUE 값을 추출하
여 HashMap에 저장
    public void getParamValueMap(List<?> fileItemList, HashMap<String,
String> paramMap) throws UnsupportedEncodingException
    {
        FileItem             fileItem = null;                // file item

        // request 파싱 처리 결과 리스트에서 파일 데이터를 제외한 일반 KEY/VALUE 값을 추
출하여 HashMap에 저장
```

```
        Iterator<?> iter1 = fileItemList.iterator();
        while (iter1.hasNext()) {
            fileItem = (FileItem)iter1.next();

       // 일반 Form 필드 값을 추출하여 paramMap HashMap에 저장
            if (fileItem.isFormField()) {
                String fieldName = fileItem.getFieldName();
                String fieldValue = fileItem.getString();

                // EXCEL_GLOBAL_INFO값등을 인코딩을 해서 올리므로 디코딩
함(EXCEL_GLOBAL_INFO값에 한글이 있는 경우관련)
                fieldValue = URLDecoder.decode(fieldValue, "UTF-8");
                paramMap.put(fieldName, fieldValue);

                logMsg("fieldName = [" + fieldName + "], fieldValue = [" +
fieldValue + "]");
            }
        }
    }

    public void setResponseCommonHeader(HttpServletRequest request,
HttpServletResponse response) throws UnsupportedEncodingException
    {
        String        origin = "";
        boolean        bCorsHeader = true;                // cors_header 사
용할지 여부

        browserType = getBrowserType(request);
        logMsg("browserType: " + browserType);

        // 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
        origin = request.getHeader("Origin");
        logMsg("origin = " + origin);

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

    public void setResponseHeader(HttpServletResponse response, String
saveAsFileName, long fileSize) throws UnsupportedEncodingException
    {
        response.setHeader("Content-Length", String.valueOf(fileSize));
        response.setHeader("Content-Transfer-Encoding", "binary");
        response.setHeader("Accept-Ranges", "bytes");
        response.setHeader("Set-Cookie", "fileDownload=true; path=/");

        /*
        // "Content-disposition: attachment"은
        // 브라우저 인식 파일확장자를 포함하여 모든 확장자의 파일들에 대해, 다운로드시 무조건 "파일
다운로드" 대화상자가 뜨도록 하는 헤더속성이라 할 수 있다.
        if(userAgent.indexOf("Safari") > -1) {
            response.setHeader("Content-Disposition", "attachment;
filename=" + excelFileName);
        } else {
            response.setHeader("Content-Disposition", "attachment;
filename*=UTF-8''" + excelFileName);
        }
        */

        if (browserType.contains("IE")) {
            saveAsFileName = URLEncoder.encode(saveAsFileName,
"UTF-8").replaceAll("\\+", "%20");
            response.setHeader("Content-Disposition", "attachment;filename="
+ saveAsFileName + ";");
        }
        else if (browserType.contains("FIREFOX")) {
            saveAsFileName = new String(saveAsFileName.getBytes("UTF-8"),
"ISO-8859-1");
            response.setHeader("Content-Disposition", "attachment;
filename=\"" + saveAsFileName + "\"");
        }
        else if (browserType.contains("OPERA")) {
            saveAsFileName = new String(saveAsFileName.getBytes("UTF-8"),
"ISO-8859-1");
            response.setHeader("Content-Disposition", "attachment;
filename=\"" + saveAsFileName + "\"");
        }
        else if (browserType.contains("CHROME")) {
            saveAsFileName = new String(saveAsFileName.getBytes("UTF-8"),
"ISO-8859-1");
            response.setHeader("Content-Disposition", "attachment;
filename=\"" + saveAsFileName + "\"");
        }
        else if (browserType.contains("SAFARI")) {
            response.setHeader("Content-Disposition", "attachment;
filename=" + saveAsFileName);
        }
    }

    public String getBrowserType(HttpServletRequest request) {
        String browser = "";
        String userAgent = request.getHeader("User-Agent");
        logMsg("user-agent:" + userAgent);

        if (userAgent.indexOf("Trident") > 0 || userAgent.indexOf("MSIE") >
0) {
             browser = "IE";
        }
        else if (userAgent.indexOf("Opera") > 0) {
             browser = "OPERA";
        }
        else if (userAgent.indexOf("Firefox") > 0) {
             browser = "FIREFOX";
        }
        else if (userAgent.indexOf("Safari") > 0) {
             if (userAgent.indexOf("Chrome") > 0) {
                  browser = "CHROME";
             }
             else {
                  browser = "SAFARI";
             }
        }

        return browser;
    }

    public String getContextRootPath(HttpServletRequest request)
    {
        String contextRootDirAbsolutePath = "";
        String resPath = "";
        String temPath = "";
        ServletContext context = request.getServletContext();

        // get context real path
        contextRootDirAbsolutePath = context.getRealPath("/");

        // Spring Boot에서는 getRealPath()로 값을 구할 수 없으므로 다른방식으로 구해줌
        if (contextRootDirAbsolutePath == null) {
            temPath =
XDrmUtil.class.getProtectionDomain().getCodeSource().getLocation().getPath()
;

       // 앞에 붙은 file:/를 삭제
            resPath = temPath.substring(temPath.indexOf(":") + 2);

            resPath = resPath.substring(0, resPath.lastIndexOf("/"));
            resPath = resPath.substring(0, resPath.lastIndexOf("/"));
            resPath = resPath.substring(0, resPath.lastIndexOf("/"));
            resPath = resPath.substring(0, resPath.lastIndexOf(".")); // 폴
더 이름으로 사용하기 위해 파일 확장자 제거
            resPath += "/";

            contextRootDirAbsolutePath = resPath;
        }

        return contextRootDirAbsolutePath;
    }

    // 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;
    }

  // 임시 파일 이름 생성하여 리턴
    private String getRandomFileName()
    {
        return java.util.UUID.randomUUID().toString().replace("-", "");
    }

  // 파일 절대 경로를 이용하여 디렉토리 생성
    public void makeDirUsingFilePath(String fileAbsolutePath)
    {
        File     oFile = new File(fileAbsolutePath);
        File    oDir = oFile.getParentFile();

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

  // 파일 절대 경로에 해당하는 파일 삭제
    public boolean deleteFile(String fileAbsolutePath)
    {
        boolean bDeleteSuccess = false;
        File     file;

        try {
            file = new File(fileAbsolutePath);

        // 파일이 존재할 경우, 파일 삭제 처리
            if (file.exists()) {
                bDeleteSuccess = file.delete();
            }
        }
        catch (Exception e) {
            logMsg("deleteUploadFile e = " + e.getMessage());
        }
        catch (Throwable ex) {
            logMsg("deleteUploadFile ex = " + ex.getMessage());
        }

        return bDeleteSuccess;
    }

    // temp 폴더 파일 삭제 되었는지 체크 후 로그 출력
    public void checkTempFileDeleted(FileItem fileItem)
    {
        String    tempFileAbsolutePath = "";
        File     tempFile = null;
        String     tempFileName;

        DeferredFileOutputStream dfos = null;

        try {
            dfos = (DeferredFileOutputStream)fileItem.getOutputStream();

            tempFileName = dfos.getFile().getName(); // temp 폴더 파일의 이름 구
하기
            tempFileAbsolutePath = tempDirAbsolutePath + File.separator +
tempFileName;

            tempFile = new File(tempFileAbsolutePath);
            if (tempFile.exists()) {
          // 파일이 존재할 경우 삭제 실패 한 것
                logMsg("Failed to delete file in the temp folder, path = " +
tempFileAbsolutePath);
            }
            else {
                logMsg("Successfully delete files in the temp folder, path =
" + tempFileAbsolutePath);
            }
        }
        catch (Exception e) {
            logMsg("checkTempFile e = " + e.getMessage());
        }
        catch (Throwable ex) {
            logMsg("checkTempFile ex = " + ex.getMessage());
        }
        finally {
            if (dfos != null) {
                try { dfos.close(); }
                catch (Exception ignore) {}
            }
        }
    }

    // FileItem에 대한 파일 데이터를 파일로 저장하고, FileItem 값 삭제 및 저장된 파일 경
로 반환
    public String handleUploadFile(FileItem fileItem, HashMap<String,
String> paramMap) throws Exception, Throwable
    {
        // FileItem의 파일 데이터를 저장할 파일 절대 경로
        String        saveFileAbsolutePath = "";

        // FileItem에 저장된 HTML FILE NAME을 구함
        String         filePath = fileItem.getName();

        fileName = FilenameUtils.getName(filePath);

        String         fileNameExt = FilenameUtils.getExtension(fileName);

        logMsg("filePath = " + filePath);
        logMsg("fileName = " + fileName);
        logMsg("fileNameExt = " + fileNameExt);

        logMsg("getContentType = " + fileItem.getContentType());
        logMsg("getSize = " + fileItem.getSize());

    // 파일 저장 디렉토리 절대 경로(saveBaseDirAbsolutePath) 및 랜덤 파일 이름을 생성하
여 저장할 파일 경로를 설정
        saveFileAbsolutePath = saveBaseDirAbsolutePath + File.separator +
getRandomFileName();
        logMsg("saveFileAbsolutePath = " + saveFileAbsolutePath);

    // 저장 파일 절대 경로를 이용하여 디렉토리 생성
        makeDirUsingFilePath(saveFileAbsolutePath);

    // 파일에 대한 fileItem 값을 파일에 저장, 오류 발생시 5회 반복
        while (true) {
            int    retryCount = 0;

            try {
                // FileItem의 파일 데이터를 파일로 저장
                fileItem.write(new File(saveFileAbsolutePath));
                break;
            }
            catch (Exception e) {
                logMsg("Exception Msg = " + e.getMessage());
                retryCount++;

                if (retryCount > 5) {
                    logMsg("Fail to wirte a file");
                    errorMsg = "Fail to wirte a file";

           // 파일이 존재할 경우, 파일 삭제 처리
                    deleteFile(saveFileAbsolutePath);

                    return null;
                }
                else {
                    try { Thread.sleep(1000); }
                    catch (InterruptedException ignore) { }
                    continue;
                }
            }
        }

        logMsg("Success To Write File");

        // File 아이템의 파일 데이터 값 삭제 처리
        fileItem.delete();

        // temp 폴더의 임시파일 삭제 유무 로그 출력
        checkTempFileDeleted(fileItem);

     // 저장된 파일 절대 경로 반환
        return saveFileAbsolutePath;
    }
}
```