

```

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.nio.channels.FileChannel;
import java.nio.charset.Charset;

import com.dlxn.core.web.WebConfiguration;

/**
 * @author zsf
 *
 * 资源操作工具类
 */
public class IOUtil {
    // 表示从本地上传
    public static final byte UPLOAD_BY_LOCAL = 0x01;
    // 直接流上传
    public static final byte UPLOAD_BY_INS = 0x02;
    // 网络下载上传
    public static final byte UPLOAD_BY_URL = 0x03;

    /**
     * 保存文件
     */
    public static boolean uploadFile(String fileName, InputStream
ins) {
        boolean errorCode = true;
        if (ins == null) {
            return false;
        }
        fileName = IOUtil.parseFilePath(fileName);
        File DBFFile = new File(fileName);
        byte[] bytes = new byte[4 * 1024];
        try {
            while (!DBFFile.exists()) {
                File parentFile = DBFFile.getParentFile();
                while (!parentFile.exists()) {
                    parentFile.mkdirs();
                }
                DBFFile.createNewFile();
            }
            FileOutputStream fileOutput = null;
            fileOutput = new FileOutputStream(DBFFile);
            int length = 0;
            ;
            while (length >= 0) {
                length = ins.read(bytes, 0, 1024);
                if (length >= 0) {

```

```

        fileOutput.write(bytes, 0, length);
    }

    }
    ins.close();
    fileOutput.flush();
    fileOutput.close();

    } catch (Exception e) {
        e.printStackTrace();
        errorCode = false;
    }
    return errorCode;
}

public static void copyFileNew(String oldPath, String newPath)
{
    try {
        FileChannel in = new FileInputStream(oldPath).getChannel();
        FileChannel out = new FileOutputStream(newPath).getChannel();

        in.transferTo(0, in.size(), out);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

/**
 *
 *
 * @param oldPath
 *      □
 * @param newPath
 *      .
 */
public static void copyFile(String oldPath, String newPath) {
    File f = new File(oldPath);
    if (!f.exists())
        return;
    FileOutputStream fileOutput = null;
    FileInputStream fileinput = null;
    byte[] bytes = new byte[4 * 1024];
    File nf = new File(newPath);
    try {
        if (!nf.exists()) {
            File pf = nf.getParentFile();
            while (!pf.exists()) {
                pf.mkdirs();
            }
            pf.createNewFile();
        }
        fileinput = new FileInputStream(f);
        fileOutput = new FileOutputStream(nf);
        int length = 0;
        while (length >= 0) {
            length = fileinput.read(bytes, 0, 1024);
            if (length >= 0) {

```

```

        fileOutput.write(bytes, 0, length);
    }

    }

    } catch (Exception e) {
        copyFile(oldPath, newPath);
        e.printStackTrace();
    } finally {
        try {
            if (fileinput != null)
                fileinput.close();
            if (fileOutput != null) {
                fileOutput.flush();
                fileOutput.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

/**
 * 将指定文件输出至输出流
 *
 * @param fileFullName
 *         完整物理路径
 * @param out
 * @return
 * @throws IOException
 */
public static boolean fileOutput(String fileFullName, OutputStream out)
    throws IOException {
    BufferedInputStream bis = null;
    BufferedOutputStream bos = null;
    try {
        bis = new BufferedInputStream(new FileInputStream(fileFullName));
        bos = new BufferedOutputStream(out);

        byte[] buff = new byte[4096];
        int bytesRead;

        while (-1 != (bytesRead = bis.read(buff, 0, buff.length))) {
            bos.write(buff, 0, bytesRead);
        }

    } catch (FileNotFoundException fnfex) {
        return false;
    } catch (final IOException e) {
        // System.out.println ( "IOException." + e );
    } finally {
        if (bis != null)
            bis.close();
        if (bos != null)

```

```

        bos.close();
    }
    return true;
}

/*
 * 得到文件的内容
 */
public static OutputStream getFile(String fileName, OutputStream
am ops) {
    if (ops == null)
        return null;
    fileName = IOUtil.parseFilePath(fileName);
    File DBFFile = new File(fileName);

    byte[] bytes = new byte[4 * 1024];
    try {
        FileInputStream fileInput = null;
        fileInput = new FileInputStream(DBFFile);
        int length = 0;
        ;
        while (length >= 0) {
            length = fileInput.read(bytes, 0, 1024);
            if (length >= 0) {
                ops.write(bytes, 0, length);
            }
        }
        fileInput.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return ops;
}

/*
 * 创建文件目录()
 */
public static boolean createSingleFolder(String calalogName) {
    boolean errorCode = false;
    calalogName = IOUtil.parseFilePath(calalogName);
    File file = new File(calalogName);
    errorCode = file.mkdir();
    return errorCode;
}

/*
 * 创建文件目录()
 */
public static boolean createMulFolder(String calalogName) {
    boolean errorCode = false;
    calalogName = IOUtil.parseFilePath(calalogName);
    File file = new File(calalogName);
    errorCode = file.mkdirs();
    return errorCode;
}

```

```

/*
 * 创建文件(假定文件的父亲目录已经存在) @param fileName 文件的全路径名
 */
public static boolean createSingleFile(String fileName) {
    boolean errorCode = true;
    /* 判断该文件的目录是否存在 */
    if (fileName == null)
        return false;

    int lastSeparator = fileName.lastIndexOf(File.separator);
    if (lastSeparator != -1) {
        String parentCal = fileName.substring(0, lastSeparator
);
        File file = new File(parentCal);
        if (!file.isDirectory()) {
            createMulFolder(fileName.substring(0, lastSeparato
r));
        }
    }

    File file = new File(IOUtil.parseFilePath(fileName));
    try {
        file.createNewFile();
    } catch (IOException e) {
        e.printStackTrace();
        errorCode = false;
    }
    return errorCode;
}

/*
 * 删除目录 @param 被删除的目录
 */
public static boolean delFolder(File calalog) {
    boolean errorCode = true;
    if (!calalog.isDirectory()) {
        calalog.delete();
    } else {
        File[] entries = calalog.listFiles();
        int sz = entries.length;
        for (int i = 0; i < sz; i++) {
            if (entries[i].isDirectory()) {
                delFolder(entries[i]);
            } else {
                entries[i].delete();
            }
        }
        calalog.delete();
    }
    return errorCode;
}

// TO DO
public static boolean dragFolder(String oldName, String newName) {
    return dragFile(oldName, newName);
}

```

```

// TO DO
public static boolean dragFile(String oldName, String newName)
{
    boolean errorCode = false;
    oldName = IOUtil.parseFilePath(oldName);
    newName = IOUtil.parseFilePath(newName);
    File oldFile = new File(oldName);
    File newFile = new File(newName);
    errorCode = oldFile.renameTo(newFile);
    return errorCode;
}

/*
 * 通过文件系统得到文件的内容
 */
public static String getXmlFile(String path) {
    StringBuffer fileStringBuffer = new StringBuffer();
    path = IOUtil.parseFilePath(path);
    try {
        FileReader fis = new FileReader(path);
        BufferedReader bReader = new BufferedReader(fis);
        String tempLine = null;
        do {
            tempLine = bReader.readLine();
            fileStringBuffer.append(tempLine);
        } while (tempLine != null && tempLine.length() > 0);

    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return fileStringBuffer.toString();
}

/*
 *
 * @author zsf
 */
public static String parseFilePath(String filePath) {
    if (filePath == null)
        return null;
    char[] pathCharArray = filePath.toCharArray();
    StringBuffer pathBuffer = new StringBuffer();
    Character char1 = new Character('\\');
    Character char2 = new Character('/');

    for (int i = 0; i < pathCharArray.length; i++) {
        if (char1.equals(new Character(pathCharArray[i]))
            || char2.equals(new Character(pathCharArray[i]
))) {
            pathBuffer.append(File.separator);
        } else {
            pathBuffer.append(pathCharArray[i]);
        }
    }
    // filePath=filePath.replaceAll("/",File.separator);
}

```

```

        // filePath=filePath.replaceAll("\\\\",File.separator);
        return pathBuffer.toString();
    }

    /**
     * 提供删除目录和文件
     *
     * @param dir
     * @throws IOException
     */
    public static boolean deleteDirectory(File dir) throws IOExcep
tion {
        boolean errorCode = false;
        if ((dir == null) || !dir.isDirectory()) {
            throw new IllegalArgumentException("Argument " + dir
                + " is not a directory. ");
        }
        File[] entries = dir.listFiles();
        int sz = entries.length;
        for (int i = 0; i < sz; i++) {
            if (entries[i].isDirectory()) {
                deleteDirectory(entries[i]);
            } else {
                errorCode = entries[i].delete();
                if (!errorCode)
                    return errorCode;
            }
        }
        errorCode = dir.delete();
        return errorCode;
    }

    /**
     * 重命名文件的名字
     */
    public static boolean reFileRename(String oldPath, String newP
ath) {
        boolean errorCode = false;
        oldPath = IOUtil.parseFilePath(oldPath);
        newPath = IOUtil.parseFilePath(newPath);
        File oldFile = new File(oldPath);
        File newFile = new File(newPath);
        if(newFile.exists())
            newFile.delete();
        else{
            newFile.getParentFile().mkdirs();
        }
        errorCode = oldFile.renameTo(newFile);
        return errorCode;
    }

    public static void main(String args[]){
        IOUtil.reFileRename("D:/1218549349921.flv","E:/1.flv");

        IOUtil.reFileRename("U:/public/videospace/temporary/10067/
1218549349921video.jpg","E:/1.jpg");
    }
}

```

```

public static boolean deleteFile(String filepath) {
    File file = new File(filepath);
    if(file != null && file.exists())
        return file.delete();
    return false;
}

public static boolean deleteFile(File file) {
    if(file != null && file.exists())
        return file.delete();
    return false;
}

/**
 *
 * @author zsf
 *
 * TODO 要更改此生成的类型注释的模板, 请转至 窗口 — 首选项 — Java —
代码样式 — 代码模板
 */
public static File getFile(String path) {
    return new File(parseFilePath(path));
}

/**
 * 校验文件对象是否为文件而不是文件夹并当该文件不存在时自动创建一个新的文件
 *
 * @param file 文件
 * @return 返回校验结果
 * @throws Exception 不是文件类型的时候抛出的异常
 */
public static boolean isValidOrMakeNewFile(File file)throws
Exception{
    if(!file.exists()){

        File parentFile = file.getParentFile();

        if(!parentFile.exists()){
            if(!parentFile.mkdirs())
                return false;
        }
        if(!file.createNewFile()){
            return false;
        }
    }else if(!file.isFile()){
        throw new IllegalStateException(file.getAbsolutePath()
+" is not file");
    }
    return true;
}

/**
 * 将字符串保存到本地文件中(编码为UTF-8)

```



```

    * @param string 字符串
    * @param file 本地文件
    * @return 保存到本地返回结果
    * @throws Exception 保存到本地抛出的异常
    */
    public static boolean string2File(String string, File file) throws Exception{
        boolean ret = false;
        if(string == null || file == null){
            throw new NullPointerException("string or file is null");
        }
        if(!isValidateOrMakeNewFile(file))
            return ret;
        ret = string2OutputStream(string, new FileOutputStream(file));
        return ret;
    }

    /**
     * 将字符串保存到本地文件中(编码为UTF-8)
     * @param string 字符串
     * @param filePath 本地文件的物理路径
     * @return 保存到本地返回结果
     * @throws Exception 保存到本地抛出的异常
     */
    public static boolean string2File(String string, String filePath) throws Exception{
        return string2File(string, new File(filePath));
    }

    /**
     * 将字符串保存到输出流中(编码为UTF-8)
     * @param string 字符串
     * @param out 输出流
     * @return 保存到输出流返回结果
     * @throws Exception 保存到输出流抛出的异常
     */
    public static boolean string2OutputStream(String string, OutputStream out) throws Exception{
        boolean ret = false;
        if(string == null || out == null){
            throw new NullPointerException("string or outputStream is null");
        }
        OutputStreamWriter osw = null;
        try{
            osw = new OutputStreamWriter(out, Charset.forName(WebConfiguration.ENCODING));
            osw.write(string);
            osw.flush();
            ret = true;
        } catch (Exception e){
            throw e;
        } finally{
            if(out != null)

```

```
        out.close();
        if(osw!=null)
            osw.close();
    }
    return ret;
}
}
```