package com.thunisoft.zhzj.common.util.ftp;

import java.awt.image.BufferedImage;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.util.Properties;

import javax.imageio.ImageIO;

import javax.imageio.stream.FileImageInputStream;

import org.apache.commons.io.IOUtils;

import org.apache.commons.net.ftp.FTPClient;

import org.apache.commons.net.ftp.FTPFile;

import org.apache.commons.net.ftp.FTPReply;

import org.apache.pdfbox.pdmodel.PDDocument;

import org.apache.pdfbox.rendering.PDFRenderer;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.itextpdf.text.pdf.PdfReader;

import com.thunisoft.artery.services.config.ArteryConfigUtil;

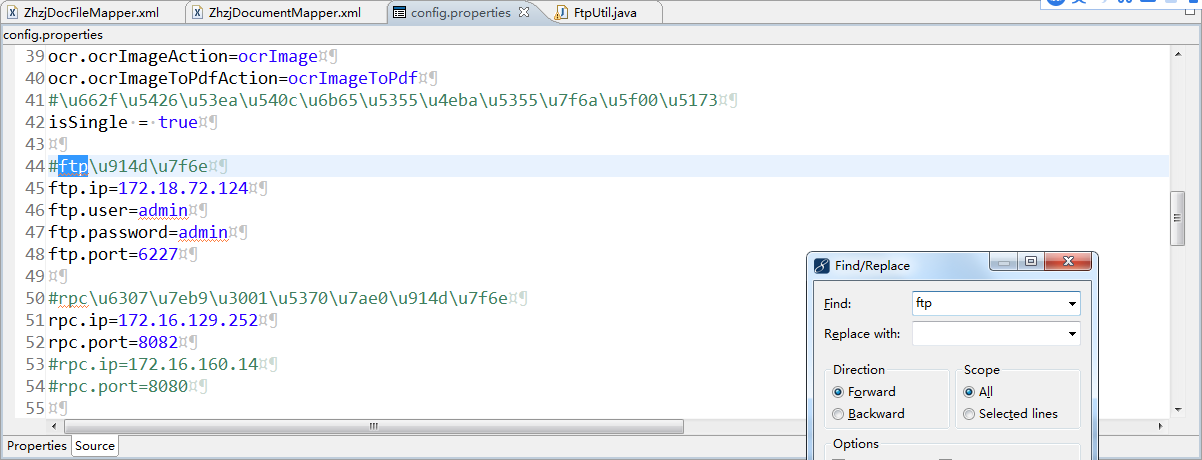
import com.thunisoft.zhzj.common.model.ZhzjDocFile;

import com.thunisoft.zhzj.common.util.pdfImage.PDF2ImageUtil;

public class FtpUtil {

// 定义日志

private static Logger logger = LoggerFactory.getLogger(FtpUtil.class);



*// ftp的Ip地址，在config.properties中，获得ftp.ip的值*

private static final String ftpIp = ArteryConfigUtil.getProperty("ftp.ip");

*// ftp的端口号，在config.properties中，获得ftp.port的值*

private static final int ftpPort = Integer.parseInt(ArteryConfigUtil

.getProperty("ftp.port"));

*// 用户名*

private static final String ftpUser = ArteryConfigUtil

.getProperty("ftp.user");

*// 密码*

private static final String ftpPassword = ArteryConfigUtil

.getProperty("ftp.password");

*// windows下载地址*

private static final String localFilePath = ArteryConfigUtil

.getProperty("downloadFtpFileToLocalPath");

*// linux下载地址*

private static final String localFilePathLinux = ArteryConfigUtil

.getProperty("downloadFtpFileToLocalPathLinux");

*// 获取连接*

private static FTPClient getConnect() {

FTPClient ftpClient = null;

try {

ftpClient = new FTPClient();

*// 连接FTP服务器*

ftpClient.connect(ftpIp, ftpPort);

*// 登录FTP服务器*

ftpClient.login(ftpUser, ftpPassword);

*//设置文件传输类型*

ftpClient.setFileType(FTPClient.BINARY\_FILE\_TYPE);

ftpClient.setControlEncoding("UTF-8");

*// 验证FTP服务器是否登录成功*

*//如果reply返回230就算成功了，如果返回530密码用户名错误或当前用户无权限下面有详细的解释。*

int replyCode = ftpClient.getReplyCode();

if (!FTPReply.isPositiveCompletion(replyCode)) {

return null;

}

} catch (Exception e) {

logger.error("连接ftp异常:ip=" + ftpIp + ",port=" + ftpPort + ",user="

+ ftpUser + ",pwd=" + ftpPassword, e);

return null;

}

return ftpClient;

}

/\*\*

\* Description: 从FTP服务器读取文件

\* return byte[]

\*/

public static byte[] downloadFile(String filename, String ftpPath) {

// 返回的数组

byte[] temp = null;

FTPClient ftpClient = null;

InputStream input = null;

try {

ftpClient = getConnect();

if (ftpClient == null) {

return null;

}

// 切换FTP目录

boolean exist = ftpClient.changeWorkingDirectory(ftpPath);

if (exist) {

FTPFile[] ftpFiles = ftpClient.listFiles();

for (FTPFile file : ftpFiles) {

if (filename.equalsIgnoreCase(file.getName())) {

input = ftpClient.retrieveFileStream(filename);

if (input != null) {

temp = IOUtils.toByteArray(input);

}

break;

}

}

}

} catch (Exception e) {

logger.error("ftp下载文件异常", e);

return null;

} finally {

IOUtils.closeQuietly(input);

if (ftpClient != null) {

try {

ftpClient.logout();

} catch (IOException e) {

logger.error("ftp退出异常", e);

}

}

}

return temp;

}

public static InputStream downloadFileAsStream(String filename,

String ftpPath) {

InputStream in = null;

byte[] bytes = downloadFile(filename, ftpPath);

in = new ByteArrayInputStream(bytes);

return in;

}

/\*\*

\* 上传ftp

\* @param fileName

\* @param filePath

\* @param upFile

\*/

public static boolean uploadFile(String fileName, String filePath,

File upFile) {

boolean result = false;

FTPClient ftpClient = null;

InputStream in = null;

try {

ftpClient = getConnect();

// 切换FTP目录

boolean exist = ftpClient.changeWorkingDirectory(filePath);

in = new FileInputStream(upFile);

if (exist) {

ftpClient.storeFile(fileName, in);

} else {

String[] dirs = filePath.split("\\\\");

boolean create = false;

for (String dir : dirs) {

boolean temp = ftpClient.changeWorkingDirectory(dir);

if (!temp) {

ftpClient.makeDirectory(dir);

temp = ftpClient.changeWorkingDirectory(dir);

}

create = temp;

}

if (create) {

ftpClient.storeFile(fileName, in);

}

}

result = true;

} catch (Exception e) {

logger.error(

"上传文件失败fileName=" + fileName + ",filePath=" + filePath, e);

return false;

} finally {

IOUtils.closeQuietly(in);

if (ftpClient != null) {

try {

ftpClient.logout();

ftpClient.disconnect();

} catch (IOException e) {

logger.error("ftp退出异常", e);

}

}

}

return result;

}

/\*\*

\* 上传ftp

\* @param fileName

\* @param filePath

\* @param upFile

\* @return boolean

\*/

public static boolean uploadFile(String fileName, String filePath,

InputStream upFile) {

boolean result = false;

FTPClient ftpClient = null;

try {

ftpClient = getConnect();

// 切换FTP目录

boolean exist = ftpClient.changeWorkingDirectory(filePath);

if (exist) {

ftpClient.storeFile(fileName, upFile);

} else {

String[] dirs = filePath.split("\\\\");

boolean create = false;

for (String dir : dirs) {

boolean temp = ftpClient.changeWorkingDirectory(dir);

if (!temp) {

ftpClient.makeDirectory(dir);

temp = ftpClient.changeWorkingDirectory(dir);

}

create = temp;

}

if (create) {

ftpClient.storeFile(fileName, upFile);

}

}

result = true;

} catch (Exception e) {

logger.error(

"上传文件失败fileName=" + fileName + ",filePath=" + filePath, e);

return false;

} finally {

IOUtils.closeQuietly(upFile);

if (ftpClient != null) {

try {

ftpClient.logout();

ftpClient.disconnect();

} catch (IOException e) {

logger.error("ftp退出异常", e);

}

}

}

return result;

}

/\*\*

\* 删除ftp文件

\*/

public static boolean deletePdf(String fileName, String filePath) {

boolean result = false;

FTPClient ftpClient = null;

try {

ftpClient = getConnect();

// 切换FTP目录

boolean exist = ftpClient.changeWorkingDirectory(filePath);

if (exist) {

ftpClient.deleteFile(fileName);

}

} catch (Exception e) {

logger.error("删除FTP的pdf文件失败", e);

}

return result;

}

/\*\*

\* 删除ftp该案件下面的所有文件

\*/

public static boolean deleteAllFile(String ftpPath) {

boolean result = false;

FTPClient ftpClient = null;

try {

ftpClient = getConnect();

FTPFile[] files = ftpClient.listFiles(ftpPath);

for (FTPFile f : files) {

String path = ftpPath + File.separator + f.getName();

if (f.isFile()) {

// 是文件就删除文件

ftpClient.deleteFile(path);

} else if (f.isDirectory()) {

deleteAllFile(path);

}

}

} catch (Exception e) {

logger.error("删除FTP文件夹下面的文件失败", e);

}

return result;

}

/\*\*

\* 删除ftp文件夹

\*/

public static boolean removeDir(String filePath) {

boolean result = false;

FTPClient ftpClient = null;

try {

ftpClient = getConnect();

// 删除文件夹（为空）

ftpClient.removeDirectory(filePath);

} catch (Exception e) {

logger.error("删除FTP的文件夹失败", e);

}

return result;

}

/\*\*

\* 下载ftp文件到本地

\* @param filename

\* @param ftpPath

\* @return

\*/

public static void downloadFileToLocal(String filename, String ftpPath,

String docId) {

// 下载的地址

String localPath = "";

// 返回的数组

byte[] temp = null;

FTPClient ftpClient = null;

InputStream input = null;

OutputStream outputStream = null;

try {

ftpClient = getConnect();

if (ftpClient == null) {

return;

}

// 切换FTP目录

boolean exist = ftpClient.changeWorkingDirectory(ftpPath);

if (exist) {

FTPFile[] ftpFiles = ftpClient.listFiles();

for (FTPFile file : ftpFiles) {

if (filename.equalsIgnoreCase(file.getName())) {

// 判断系统是windows还是linux

Properties prop = System.getProperties();

String os = prop.getProperty("os.name");

if (os != null

&& os.toLowerCase().indexOf("linux") > -1) {

localPath = localFilePathLinux + File.separator

+ docId;

File entryDir = new File(localPath);

//如果文件夹路径不存在，则创建文件夹

if (!entryDir.exists() || !entryDir.isDirectory()) {

entryDir.mkdirs();

}

File locaFile = new File(localPath + File.separator

+ file.getName());

outputStream = new FileOutputStream(localPath

+ File.separator + filename);

} else {

localPath = localFilePath + "/" + docId;

File entryDir = new File(localPath);

//如果文件夹路径不存在，则创建文件夹

if (!entryDir.exists() || !entryDir.isDirectory()) {

entryDir.mkdirs();

}

File locaFile = new File(localPath + "/"

+ file.getName());

outputStream = new FileOutputStream(localPath + "/"

+ filename);

}

ftpClient.retrieveFile(filename, outputStream);

outputStream.flush();

return;

}

}

}

} catch (Exception e) {

logger.error("ftp下载文件至本地异常", e);

return;

} finally {

IOUtils.closeQuietly(outputStream);

if (ftpClient != null) {

try {

ftpClient.logout();

} catch (IOException e) {

logger.error("ftp退出异常", e);

}

}

}

}

/\*\*\*

\* PDF文件转PNG图片，全部页数

\*

\* @param PdfFilePath

\* pdf完整路径

\* @param imgFilePath

\* 图片存放的文件夹

\* @param dpi

\* dpi越大转换后越清晰，相对转换速度越慢

\* @return

\*/

public static StringBuffer pdfzhuanImage(String PdfFilePath,

String dstImgFolder, int dpi) {

File file = new File(PdfFilePath);

StringBuffer imgFilePath = null;

PDDocument pdDocument;

try {

String imgPDFPath = file.getParent();

int dot = file.getName().lastIndexOf('.');

String imagePDFName = file.getName().substring(0, dot); // 获取图片文件名

String imgFolderPath = null;

if (dstImgFolder.equals("")) {

imgFolderPath = imgPDFPath;//

// 获取图片存放的文件夹路径

} else {

imgFolderPath = dstImgFolder;

}

if (createDirectory(imgFolderPath)) {

pdDocument = PDDocument.load(file);

PDFRenderer renderer = new PDFRenderer(pdDocument);

/\* dpi越大转换后越清晰，相对转换速度越慢 \*/

PdfReader reader = new PdfReader(PdfFilePath);

int pages = reader.getNumberOfPages();

for (int i = 0; i < pages; i++) {

String imgFilePathPrefix = imgFolderPath + "/"

+ imagePDFName;

imgFilePath = new StringBuffer();

imgFilePath.append(imgFilePathPrefix);

imgFilePath.append(".png");

File dstFile = new File(imgFilePath.toString());

BufferedImage image = renderer.renderImageWithDPI(i, dpi);

ImageIO.write(image, "png", dstFile);

}

} else {

System.out.println("PDF文档转PNG图片失败：" + "创建" + imgFolderPath

+ "失败");

}

} catch (IOException e) {

e.printStackTrace();

}

return imgFilePath;

}

//图片到byte数组

public static byte[] imagezhuanbyte(String path) {

byte[] data = null;

FileImageInputStream input = null;

try {

input = new FileImageInputStream(new File(path));

ByteArrayOutputStream output = new ByteArrayOutputStream();

byte[] buf = new byte[1024];

int numBytesRead = 0;

while ((numBytesRead = input.read(buf)) != -1) {

output.write(buf, 0, numBytesRead);

}

data = output.toByteArray();

output.close();

input.close();

} catch (FileNotFoundException ex1) {

ex1.printStackTrace();

} catch (IOException ex1) {

ex1.printStackTrace();

} finally {

IOUtils.closeQuietly(input);

}

return data;

}

private static boolean createDirectory(String folder) {

File dir = new File(folder);

if (dir.exists()) {

return true;

} else {

return dir.mkdirs();

}

}

/\*\*

\* PDF转换图片

\* @param docidcopy

\* @param ftpPath

\* @param fileName

\* @return

\* @throws Exception

\*/

public static InputStream PdfzhunhuanTP(ZhzjDocFile zhzjDocFile) throws Exception {

// 图片地址

String localPaths = "";

String localPath = "";

// 下载到本地 pdf的地址

String localPathpdf = "";

// 文件夹的名字

String docid = zhzjDocFile.getDocId();

// 要保存的文件名字

String fileName = zhzjDocFile.getId();

// 要下载的ftp地址

String ftpPath = zhzjDocFile.getDocPath();

// 文件的名字

String fileNameAll = zhzjDocFile.getId() + zhzjDocFile.getDocFileType();

// 在ftp上下载pdf图片

downloadFileToLocal(fileNameAll, ftpPath, docid);

// 判断系统是windows还是linux

Properties prop = System.getProperties();

String os = prop.getProperty("os.name");

if (os != null && os.toLowerCase().indexOf("linux") > -1) {

// 图片地址

localPath = localFilePathLinux + File.separator + docid;

// 下载到本地 pdf的地址

localPathpdf = localPath + File.separator + fileNameAll;

} else {

// 图片地址

localPaths = localFilePath + "/" + docid;

localPath = localPaths.replaceAll("//", "/");

// 下载到本地 pdf的地址

localPathpdf = localPath + "/" + fileNameAll;

}

// 进行pdf转换图片 调用服务

File pdfFile = new File(localPathpdf);

InputStream inputStreampng = PDF2ImageUtil.pdf2Image(pdfFile);

logger.info("图片转换完成.....");

File deletepdf = new File(localPathpdf);

deletepdf.delete();

return inputStreampng;

}

}

("ftp.port"));

**最后附上ftpClient.getReplyCode()的返回值的对应信息：**  
110  重新启动标记应答。在这种情况下文本是确定的，它必须是：MARK   yyyy=mmmm，其中yyyy是用户进程数据流标记，mmmm是服务器标记。             
120     服务在nnn分钟内准备好             
125     数据连接已打开，准备传送             
150     文件状态良好，打开数据连接             
200     命令成功             
202     命令未实现             
211     系统状态或系统帮助响应             
212     目录状态             
213     文件状态             
214     帮助信息，信息仅对人类用户有用             
215     名字系统类型             
220     对新用户服务准备好             
221     服务关闭控制连接，可以退出登录             
225     数据连接打开，无传输正在进行             
226     关闭数据连接，请求的文件操作成功             
227     进入被动模式             
230     用户登录             
250     请求的文件操作完成             
257     创建 "PATHNAME "             
331     用户名正确，需要口令             
332     登录时需要帐户信息             
350     请求的文件操作需要进一步命令             
421     不能提供服务，关闭控制连接             
425     不能打开数据连接             
426     关闭连接，中止传输             
450     请求的文件操作未执行             
451     中止请求的操作：有本地错误             
452     未执行请求的操作：系统存储空间不足             
500     格式错误，命令不可识别             
501     参数语法错误             
502     命令未实现             
503     命令顺序错误             
504     此参数下的命令功能未实现             
530     未登录（用户名或密码错误，1、FTP密码修改了？2、用户名/密码输入错误？先仔细检查有无输入错误   如复制的时候误复制了空格！！）             
532     存储文件需要帐户信息             
550     未执行请求的操作             
551     请求操作中止：页类型未知             
552     请求的文件操作中止，存储分配溢出             
553     未执行请求的操作：文件名不合法

byte[]字节数组转换为File文件：

/\*\*

\* 根据byte数组，生成文件

\* **@param** bfile 文件数组

\* **@param** filePath 文件存放路径

\* **@param** fileName 文件名称

\*/

**public** **static** File byte2File(**byte**[] bfile, String filePath, String fileName) {

BufferedOutputStream bos = **null**;

FileOutputStream fos = **null**;

File file = **null**;

**try** {

file = **new** File(filePath);

fos = **new** FileOutputStream(file);

bos = **new** BufferedOutputStream(fos);

bos.write(bfile);

} **catch** (Exception e) {

*logger*.error("字节转换文件错误");

} **finally** {

**try** {

**if** (bos != **null**) {

bos.close();

}

**if** (fos != **null**) {

fos.close();

}

} **catch** (Exception e) {

*logger*.error("字节转换文件错误");

}

}

**return** file;

}