问题描述：

问题涉及三个平台，源图片服务器，网络服务器，文件管理服务器。网络服务器从源图片服务器通过后台HttpConnection方式获取图片二进制流，并将二进制流转换成base64编码格式，再次通过HttpConnection方式发送到文件服务器进行存储。

第一种实现方式：

imgUrl : 为一个图片链接。

P**ublic static void save(imgUrl) {**

**URL url = new URL(imgUrl);**

**//获取图片二进制流**

**InputStream imgStream = url.openStream();**

**//将二进制流转化成base64编码格式**

**String base64ImgStr = getImageStr(imgStream);**

**//将图片信息保存到文件管理器**

**String xmlStr = “<info>”+**

**“<path>file/20140301001.jpg</path>”+**

**“<img>”+ base64ImgStr +”</img>”+**

**“</info>”**

**saveImg(xmlStr);**

**}**

**public** **static** String getImageStr(InputStream in) **throws** Exception {

**byte**[] data = **null**;

// 读取图片字节数组

**try** {

data = **new** **byte**[in.available()];

in.read(data);

in.close();

} **catch** (IOException e) {

e.printStackTrace();

}

// 对字节数组Base64编码

BASE64Encoder encoder = **new** BASE64Encoder();

// 返回Base64编码过的字节数组字符串

**return** encoder.encode(data);

}

按这种方式保存图片，文件比较小时（10K以下）时图片能完整保存，如果过大则只能保存部分，图片显示不完整。

第二种方式：

将二进制流分段读取并转成base64编码

**public** **static** **void** getImageStr (String \_path, String \_savePath) {

String savePath = \_savePath;

String path = \_path;

**int** BYTE\_SIZE = 1;

**int** SAVE\_SIZE = 1024;

**byte**[] buff = **new** **byte**[BYTE\_SIZE]; // 每次读的缓存

**byte**[] save = **new** **byte**[SAVE\_SIZE]; // 保存前缓存

BufferedInputStream bf = **null**;

FileOutputStream file;

URL url = **null**;

String imgBase64 = "";

HttpURLConnection httpUrl;

**try** {

// 对字节数组Base64编码

BASE64Encoder encoder = **new** BASE64Encoder();

url = **new** URL(path);

httpUrl = (HttpURLConnection) url.openConnection();

bf = **new** BufferedInputStream(httpUrl.getInputStream());

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

**int** i = 0;

**while** (bf.read(buff) != -1) { // 一个字节一个字节读

save[i] = buff[0];

**if** (i == SAVE\_SIZE - 1) { // 达到保存长度时开始保存

imgBase64 += encoder.encode(save);

file.write(save, 0, SAVE\_SIZE);

save = **new** **byte**[SAVE\_SIZE];

i = 0;

} **else** {

i++;

}

}

// 最后这段如果没达到保存长度，需要把前面的保存下来

**if** (i > 0) {

imgBase64 += encoder.encode(save);

file.write(save, 0, i - 1);

}

httpUrl.disconnect();

file.close();

} **catch** (MalformedURLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **finally** {

**try** {

bf.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

第三种方式：

将所有的二进制流同时保存，然后一起进行base64编码。

**private** **static** String getImgToBase64(String \_path) {

String path = \_path;

**int** BYTE\_SIZE = 1;

**int** SAVE\_SIZE = 1024;

**byte**[] buff = **new** **byte**[BYTE\_SIZE]; // 每次读的缓存

**byte**[] save = **new** **byte**[SAVE\_SIZE]; // 保存前缓存

**byte**[] imgByte = **null**;

String imgStr = **null**;

BufferedInputStream bf = **null**;

URL url = **null**;

HttpURLConnection httpUrl;

List<**byte**[]> byteList = **null**;

**try** {

byteList = **new** ArrayList<**byte**[]>();

// 对字节数组Base64编码

BASE64Encoder encoder = **new** BASE64Encoder();

url = **new** URL(path);

httpUrl = (HttpURLConnection) url.openConnection();

bf = **new** BufferedInputStream(httpUrl.getInputStream());

**int** i = 0;

**while** (bf.read(buff) != -1) { // 一个字节一个字节读

save[i] = buff[0];

**if** (i == SAVE\_SIZE - 1) { // 达到保存长度时开始保存

// 返回Base64编码过的字节数组字符串

byteList.add(save);

save = **new** **byte**[SAVE\_SIZE];

i = 0;

} **else** {

i++;

}

}

// 最后这段如果没达到保存长度，需要把前面的保存下来

**if** (i > 0) {

// 返回Base64编码过的字节数组字符串

byteList.add(save);

}

**int** length = byteList.size();

imgByte = **new** **byte**[SAVE\_SIZE \* (length - 1) + i];

**for** (**int** j = 0; j < byteList.size() - 1; j++) {

**byte**[] temp = byteList.get(j);

System.*arraycopy*(temp, 0, imgByte, j \* SAVE\_SIZE, SAVE\_SIZE);

}

**if** (i > 0) {

System.*arraycopy*(save, 0, imgByte, (length - 1) \* SAVE\_SIZE, i);

}

imgStr = encoder.encode(imgByte);

httpUrl.disconnect();

} **catch** (MalformedURLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **finally** {

**try** {

bf.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

**return** imgStr;

}

第三种方式测试成功

**private** **static** String saveImg(String xml) **throws** ServletException,

IOException {

URL postUrl = **new** URL("");

HttpURLConnection connection = (HttpURLConnection) postUrl

.openConnection();

connection.setDoOutput(**true**);

connection.setDoInput(**true**);

connection.setRequestMethod("POST");

connection

.setRequestProperty("Content-Type", "text/xml; charset=utf-8");

connection

.setRequestProperty(

"User-Agent",

"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; "

+ ".NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; .NET4.0C; .NET4.0E)");

connection.setRequestProperty("Content-Length",

String.*valueOf*(xml.getBytes().length));

connection.connect();

OutputStream out = connection.getOutputStream();

out.write(xml.getBytes());

out.flush();

out.close(); // 到此时服务器已经收到了完整的http

// request了，而在readContentFromPost()函数里，要等到下一句服务器才能收到http请求。

BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(

connection.getInputStream()));

String line;

StringBuffer sb = **new** StringBuffer();

**while** ((line = reader.readLine()) != **null**) {

sb.append(line);

}

reader.close();

connection.disconnect();

**return** sb.toString();

}

引用包

**import** sun.misc.BASE64Encoder;