目录

[第一部分 Word文档自动追加表格内容 2](#_Toc301607591)

[第二部分 java jacob 操作word 文档，进行写操作，如生成表格，添加 图片 3](#_Toc301607592)

[第三部分 28](#_Toc301607593)

# 第一部分 Word文档自动追加表格内容

今天在做一个自动化生成SDD文档的小工具，通过Word的模板，前台通过Flex填入数据，最后将Word文档填写好。

以下是一部分代码，关于表格自动追加表格内容

**Java代码**

1. **import** java.util.ArrayList;
3. **import** com.heavenlake.wordapi.Document;
5. **public** **class** test
6. {
7. **public** test()
8. {
9. Document doc = **null**;
10. **try** {
11. doc = **new** Document();
12. doc.newDoc("e:/test4.doc");
13. doc.insertln("测试文档");
14. doc.insertln("天池软件产品列表");
15. //doc.saveAs("e:/test2.doc");
16. ArrayList tableData = **new** ArrayList(3);
17. String[] title = {
18. "序号", "代码", "名称","网址"};
19. tableData.add(title);
20. String[] field1 = {
21. "1", "751002", "天池软件自动化生产平台","www.heavenlake.com"};
22. tableData.add(field1);
23. String[] field2 = {
24. "2", "751004", "产品数据管理系统","pdm.heavenlake.com"};
25. tableData.add(field2);
27. doc.insert(tableData,"流行型");
29. doc.insertln();

32. ArrayList tableData1 = **new** ArrayList(3);
34. String[] field3 = {
35. "1", "751002", "天池软件自动化生产平台","www.heavenlake.com"};
36. tableData1.add(field3);
37. String[] field4 = {
38. "2", "751004", "产品数据管理系统","pdm.heavenlake.com"};
39. tableData1.add(field4);
40. doc.replaceTable(1, 4, tableData1);
42. } **catch** (Exception e)
43. {
44. e.printStackTrace();
45. } **finally**
46. {
47. **try**
48. {
49. **if** (doc != **null**)
50. doc.close(**true**);
51. } **catch** (Exception e)
52. {
53. e.printStackTrace();
54. }
56. }
58. }
60. **public** **static** **void** main(String[] args)
61. {
62. test test1 = **new** test();
63. }
65. }

# 第二部分 [java jacob 操作word 文档，进行写操作，如生成表格，添加 图片](http://blog.csdn.net/jixiuffff/article/details/5618287)

jacob-1.15-M3.zip

jacob-1.15-M3-x86.dll copy 到c://windows/system32

引入jacob.jar

示例代码

[view plain](http://blog.csdn.net/jixiuffff/article/details/5618287)

1. **import** java.io.File;
2. **import** com.jacob.activeX.ActiveXComponent;
3. **import** com.jacob.com.Dispatch;
4. **import** com.jacob.com.Variant;
5. **class** WordBean {
6. // 代表一个word 程序
7. **private** ActiveXComponent MsWordApp = **null**;
8. // 代表进行处理的word 文档
9. **private** Dispatch document = **null**;
10. **public** WordBean() {
11. // Open Word if we/'ve not done it already
12. **if** (MsWordApp == **null**) {
13. MsWordApp = **new** ActiveXComponent("Word.Application");
14. }
15. }
16. // 设置是否在前台打开 word 程序 ，
17. **public** **void** setVisible(**boolean** visible) {
18. MsWordApp.setProperty("Visible", **new** Variant(visible));
19. // 这一句作用相同
20. // Dispatch.put(MsWordApp, "Visible", new Variant(visible));
21. }
22. // 创建一个新文档
23. **public** **void** createNewDocument() {
24. // Find the Documents collection object maintained by Word
25. // documents表示word的所有文档窗口，（word是多文档应用程序）
26. Dispatch documents = Dispatch.get(MsWordApp, "Documents").toDispatch();
27. // Call the Add method of the Documents collection to create
28. // a new document to edit
29. document = Dispatch.call(documents, "Add").toDispatch();
30. }
31. // 打开一个存在的word文档,并用document 引用 引用它
32. **public** **void** openFile(String wordFilePath) {
33. // Find the Documents collection object maintained by Word
34. // documents表示word的所有文档窗口，（word是多文档应用程序）
35. Dispatch documents = Dispatch.get(MsWordApp, "Documents").toDispatch();
36. document = Dispatch.call(documents, "Open", wordFilePath,
37. **new** Variant(**true**)/\* 是否进行转换ConfirmConversions \*/,
38. **new** Variant(**false**)/\* 是否只读 \*/).toDispatch();
39. // document = Dispatch.invoke(documents, "Open", Dispatch.Method,
40. // new Object[] { wordFilePath, new Variant(true),
41. // new Variant(false)
42. // }, new int[1]).toDispatch();
43. }
44. // 向 document 中插入文本内容
45. **public** **void** insertText(String textToInsert) {
46. // Get the current selection within Word at the moment.
47. // a new document has just been created then this will be at
48. // the top of the new doc 获得选 中的内容，如果是一个新创建的文件，因里面无内容，则光标应处于文件开头处
49. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch();
50. // 取消选中,应该就是移动光标 ，否则 新添加的内容会覆盖选中的内容
51. Dispatch.call(selection, "MoveRight", **new** Variant(1), **new** Variant(1));
52. // Put the specified text at the insertion point
53. Dispatch.put(selection, "Text", textToInsert);
54. // 取消选中,应该就是移动光标
55. Dispatch.call(selection, "MoveRight", **new** Variant(1), **new** Variant(1));
56. }
57. // 向文档中添加 一个图片，
58. **public** **void** insertJpeg(String jpegFilePath) {
59. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch();
60. Dispatch image = Dispatch.get(selection, "InLineShapes").toDispatch();
61. Dispatch.call(image, "AddPicture", jpegFilePath);
62. }
63. // 段落的处理,插入格式化的文本
64. **public** **void** insertFormatStr(String text) {
65. Dispatch wordContent = Dispatch.get(document, "Content").toDispatch(); // 取得word文件的内容
66. Dispatch.call(wordContent, "InsertAfter", text);// 插入一个段落到最后
67. Dispatch paragraphs = Dispatch.get(wordContent, "Paragraphs")
68. .toDispatch(); // 所有段落
69. **int** paragraphCount = Dispatch.get(paragraphs, "Count").changeType(
70. Variant.VariantInt).getInt();// 一共的段落数
71. // 找到刚输入的段落，设置格式
72. Dispatch lastParagraph = Dispatch.call(paragraphs, "Item",
73. **new** Variant(paragraphCount)).toDispatch(); // 最后一段（也就是刚插入的）
74. // Range 对象表示文档中的一个连续范围，由一个起始字符位置和一个终止字符位置定义
75. Dispatch lastParagraphRange = Dispatch.get(lastParagraph, "Range")
76. .toDispatch();
77. Dispatch font = Dispatch.get(lastParagraphRange, "Font").toDispatch();
78. Dispatch.put(font, "Bold", **new** Variant(**true**)); // 设置为黑体
79. Dispatch.put(font, "Italic", **new** Variant(**true**)); // 设置为斜体
80. Dispatch.put(font, "Name", **new** Variant("宋体")); //
81. Dispatch.put(font, "Size", **new** Variant(12)); // 小四
82. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch();
83. Dispatch.call(selection, "TypeParagraph");// 插入一个空行
84. Dispatch alignment = Dispatch.get(selection, "ParagraphFormat")
85. .toDispatch();// 段落格式
86. Dispatch.put(alignment, "Alignment", "2"); // (1:置中 2:靠右 3:靠左)
87. }
88. // word 中在对表格进行遍历的时候 ，是先列后行 先column 后cell
89. // 另外下标从1开始
90. **public** **void** insertTable(String tableTitle, **int** row, **int** column) {
91. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象
92. Dispatch.call(selection, "TypeText", tableTitle); // 写入标题内容 // 标题格行
93. Dispatch.call(selection, "TypeParagraph"); // 空一行段落
94. Dispatch.call(selection, "TypeParagraph"); // 空一行段落
95. Dispatch.call(selection, "MoveDown"); // 游标往下一行
96. // 建立表格
97. Dispatch tables = Dispatch.get(document, "Tables").toDispatch();
98. // int count = Dispatch.get(tables,
99. // "Count").changeType(Variant.VariantInt).getInt(); // document中的表格数量
100. // Dispatch table = Dispatch.call(tables, "Item", new Variant(
101. // 1)).toDispatch();//文档中第一个表格
102. Dispatch range = Dispatch.get(selection, "Range").toDispatch();// /当前光标位置或者选中的区域
103. Dispatch newTable = Dispatch.call(tables, "Add", range,
104. **new** Variant(row), **new** Variant(column), **new** Variant(1))
105. .toDispatch(); // 设置row,column,表格外框宽度
106. Dispatch cols = Dispatch.get(newTable, "Columns").toDispatch(); // 此表的所有列，
107. **int** colCount = Dispatch.get(cols, "Count").changeType(
108. Variant.VariantInt).getInt();// 一共有多少列 实际上这个数==column
109. System.out.println(colCount + "列");
110. **for** (**int** i = 1; i <= colCount; i++) { // 循环取出每一列
111. Dispatch col = Dispatch.call(cols, "Item", **new** Variant(i))
112. .toDispatch();
113. Dispatch cells = Dispatch.get(col, "Cells").toDispatch();// 当前列中单元格
114. **int** cellCount = Dispatch.get(cells, "Count").changeType(
115. Variant.VariantInt).getInt();// 当前列中单元格数 实际上这个数等于row
116. **for** (**int** j = 1; j <= cellCount; j++) {// 每一列中的单元格数
117. // Dispatch cell = Dispatch.call(cells, "Item", new
118. // Variant(j)).toDispatch(); //当前单元格
119. // Dispatch cell = Dispatch.call(newTable, "Cell", new
120. // Variant(j) , new Variant(i) ).toDispatch(); //取单元格的另一种方法
121. // Dispatch.call(cell, "Select");//选中当前单元格
122. // Dispatch.put(selection, "Text",
123. // "第"+j+"行，第"+i+"列");//往选中的区域中填值，也就是往当前单元格填值
124. putTxtToCell(newTable, j, i, "第" + j + "行，第" + i + "列");// 与上面四句的作用相同
125. }
126. }
127. }
128. /\*\* \*/
129. /\*\*
130. \* 在指定的单元格里填写数据
131. \*
132. \* @param tableIndex
133. \* @param cellRowIdx
134. \* @param cellColIdx
135. \* @param txt
136. \*/
137. **public** **void** putTxtToCell(Dispatch table, **int** cellRowIdx, **int** cellColIdx,
138. String txt) {
139. Dispatch cell = Dispatch.call(table, "Cell", **new** Variant(cellRowIdx),
140. **new** Variant(cellColIdx)).toDispatch();
141. Dispatch.call(cell, "Select");
142. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象
143. Dispatch.put(selection, "Text", txt);
144. }
145. /\*\* \*/
146. /\*\*
147. \* 在指定的单元格里填写数据
148. \*
149. \* @param tableIndex
150. \* @param cellRowIdx
151. \* @param cellColIdx
152. \* @param txt
153. \*/
154. **public** **void** putTxtToCell(**int** tableIndex, **int** cellRowIdx, **int** cellColIdx,
155. String txt) {
156. // 所有表格
157. Dispatch tables = Dispatch.get(document, "Tables").toDispatch();
158. // 要填充的表格
159. Dispatch table = Dispatch.call(tables, "Item", **new** Variant(tableIndex))
160. .toDispatch();
161. Dispatch cell = Dispatch.call(table, "Cell", **new** Variant(cellRowIdx),
162. **new** Variant(cellColIdx)).toDispatch();
163. Dispatch.call(cell, "Select");
164. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象
165. Dispatch.put(selection, "Text", txt);
166. }
167. // 合并两个单元格
168. **public** **void** mergeCell(Dispatch cell1, Dispatch cell2) {
169. Dispatch.call(cell1, "Merge", cell2);
170. }
171. **public** **void** mergeCell(Dispatch table, **int** row1, **int** col1, **int** row2, **int** col2) {
172. Dispatch cell1 = Dispatch.call(table, "Cell", **new** Variant(row1),
173. **new** Variant(col1)).toDispatch();
174. Dispatch cell2 = Dispatch.call(table, "Cell", **new** Variant(row2),
175. **new** Variant(col2)).toDispatch();
176. mergeCell(cell1, cell2);
177. }
178. **public** **void** mergeCellTest() {
179. Dispatch tables = Dispatch.get(document, "Tables").toDispatch();
180. **int** tableCount = Dispatch.get(tables, "Count").changeType(
181. Variant.VariantInt).getInt(); // document中的表格数量
182. Dispatch table = Dispatch.call(tables, "Item", **new** Variant(tableCount))
183. .toDispatch();// 文档中最后一个table
184. mergeCell(table, 1, 1, 1, 2);// 将table 中x=1,y=1 与x=1,y=2的两个单元格合并
185. }
186. // ========================================================
187. /\*\* \*/
188. /\*\*
189. \* 把选定的内容或光标插入点向上移动
190. \*
191. \* @param pos
192. \* 移动的距离
193. \*/
194. **public** **void** moveUp(**int** pos) {
195. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象
196. **for** (**int** i = 0; i < pos; i++) {
197. // MoveDown MoveLeft moveRight
198. // moveStart ( Dispatch.call(selection, "HomeKey", new Variant(6));
199. // )
200. // moveEnd Dispatch.call(selection, "EndKey", new Variant(6));
201. Dispatch.call(selection, "MoveUp");
202. }
203. }
204. /\*\* \*/
205. /\*\*
206. \* 从选定内容或插入点开始查找文本
207. \*
208. \* @param toFindText
209. \* 要查找的文本
210. \* @return boolean true-查找到并选中该文本，false-未查找到文本
211. \*/
212. **public** **boolean** find(String toFindText) {
213. **if** (toFindText == **null** || toFindText.equals(""))
214. **return** **false**;
215. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象
216. // 从selection所在位置开始查询
217. Dispatch find = Dispatch.call(selection, "Find").toDispatch();
218. // 设置要查找的内容
219. Dispatch.put(find, "Text", toFindText);
220. // 向前查找
221. Dispatch.put(find, "Forward", "True");
222. // 设置格式
223. Dispatch.put(find, "Format", "True");
224. // 大小写匹配
225. Dispatch.put(find, "MatchCase", "True");
226. // 全字匹配
227. Dispatch.put(find, "MatchWholeWord", "True");
228. // 查找并选中
229. **return** Dispatch.call(find, "Execute").getBoolean();
230. }
231. /\*\* \*/
232. /\*\*
233. \* 把选定选定内容设定为替换文本
234. \*
235. \* @param toFindText
236. \* 查找字符串
237. \* @param newText
238. \* 要替换的内容
239. \* @return
240. \*/
241. **public** **boolean** replaceText(String toFindText, String newText) {
242. **if** (!find(toFindText))
243. **return** **false**;
244. Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象
245. Dispatch.put(selection, "Text", newText);
246. **return** **true**;
247. }
248. **public** **void** printFile() {
249. // Just print the current document to the default printer
250. Dispatch.call(document, "PrintOut");
251. }
252. // 保存文档的更改
253. **public** **void** save() {
254. Dispatch.call(document, "Save");
255. }
256. **public** **void** saveFileAs(String filename) {
257. Dispatch.call(document, "SaveAs", filename);
258. }
259. **public** **void** closeDocument() {
260. // Close the document without saving changes
261. // 0 = wdDoNotSaveChanges
262. // -1 = wdSaveChanges
263. // -2 = wdPromptToSaveChanges
264. Dispatch.call(document, "Close", **new** Variant(0));
265. document = **null**;
266. }
267. **public** **void** closeWord() {
268. Dispatch.call(MsWordApp, "Quit");
269. MsWordApp = **null**;
270. document = **null**;
271. }
272. // 设置wordApp打开后窗口的位置
273. **public** **void** setLocation() {
274. Dispatch activeWindow = Dispatch.get(MsWordApp, "Application")
275. .toDispatch();
276. Dispatch.put(activeWindow, "WindowState", **new** Variant(1)); // 0=default
277. // 1=maximize
278. // 2=minimize
279. Dispatch.put(activeWindow, "Top", **new** Variant(0));
280. Dispatch.put(activeWindow, "Left", **new** Variant(0));
281. Dispatch.put(activeWindow, "Height", **new** Variant(600));
282. Dispatch.put(activeWindow, "width", **new** Variant(800));
283. }
284. }
285. **public** **class** JacobTest2 {
286. **public** **static** **void** createANewFileTest() {
287. WordBean wordBean = **new** WordBean();
288. // word.openWord(true);// 打开 word 程序
289. wordBean.setVisible(**true**);
290. wordBean.createNewDocument();// 创建一个新文档
291. wordBean.setLocation();// 设置打开后窗口的位置
292. wordBean.insertText("你好");// 向文档中插入字符
293. wordBean.insertJpeg("D:" + File.separator + "a.jpg"); // 插入图片
294. // 如果 ，想保存文件，下面三句
295. // word.saveFileAs("d://a.doc");
296. // word.closeDocument();
297. // word.closeWord();
298. }
299. **public** **static** **void** openAnExistsFileTest() {
300. WordBean wordBean = **new** WordBean();
301. wordBean.setVisible(**true**); // 是否前台打开word 程序，或者后台运行
302. wordBean.openFile("d://a.doc");
303. wordBean.insertJpeg("D:" + File.separator + "a.jpg"); // 插入图片(注意刚打开的word
304. // ，光标处于开头，故，图片在最前方插入)
305. wordBean.save();
306. wordBean.closeDocument();
307. wordBean.closeWord();
308. }
309. **public** **static** **void** insertFormatStr(String str) {
310. WordBean wordBean = **new** WordBean();
311. wordBean.setVisible(**true**); // 是否前台打开word 程序，或者后台运行
312. wordBean.createNewDocument();// 创建一个新文档
313. wordBean.insertFormatStr(str);// 插入一个段落，对其中的字体进行了设置
314. }
315. **public** **static** **void** insertTableTest() {
316. WordBean wordBean = **new** WordBean();
317. wordBean.setVisible(**true**); // 是否前台打开word 程序，或者后台运行
318. wordBean.createNewDocument();// 创建一个新文档
319. wordBean.setLocation();
320. wordBean.insertTable("表名", 3, 2);
321. wordBean.saveFileAs("d://table.doc");
322. wordBean.closeDocument();
323. wordBean.closeWord();
324. }
325. **public** **static** **void** mergeTableCellTest() {
326. insertTableTest();//生成d://table.doc
327. WordBean wordBean = **new** WordBean();
328. wordBean.setVisible(**true**); // 是否前台打开word 程序，或者后台运行
329. wordBean.openFile("d://table.doc");
330. wordBean.mergeCellTest();
331. }
332. **public** **static** **void** main(String[] args) {
333. // 进行测试前要保证d://a.jpg 图片文件存在
334. // createANewFileTest();//创建一个新文件
335. // openAnExistsFileTest();// 打开一个存在 的文件
336. // insertFormatStr("格式 化字符串");//对字符串进行一定的修饰
337. //insertTableTest();// 创建一个表格
338. mergeTableCellTest();// 对表格中的单元格进行合并
339. }
340. }

import java.io.File;

import com.jacob.activeX.ActiveXComponent;

import com.jacob.com.Dispatch;

import com.jacob.com.Variant;

class WordBean {

// 代表一个word 程序

private ActiveXComponent MsWordApp = null;

// 代表进行处理的word 文档

private Dispatch document = null;

public WordBean() {

// Open Word if we/'ve not done it already

if (MsWordApp == null) {

MsWordApp = new ActiveXComponent("Word.Application");

}

}

// 设置是否在前台打开 word 程序 ，

public void setVisible(boolean visible) {

MsWordApp.setProperty("Visible", new Variant(visible));

// 这一句作用相同

// Dispatch.put(MsWordApp, "Visible", new Variant(visible));

}

// 创建一个新文档

public void createNewDocument() {

// Find the Documents collection object maintained by Word

// documents表示word的所有文档窗口，（word是多文档应用程序）

Dispatch documents = Dispatch.get(MsWordApp, "Documents").toDispatch();

// Call the Add method of the Documents collection to create

// a new document to edit

document = Dispatch.call(documents, "Add").toDispatch();

}

// 打开一个存在的word文档,并用document 引用 引用它

public void openFile(String wordFilePath) {

// Find the Documents collection object maintained by Word

// documents表示word的所有文档窗口，（word是多文档应用程序）

Dispatch documents = Dispatch.get(MsWordApp, "Documents").toDispatch();

document = Dispatch.call(documents, "Open", wordFilePath,

new Variant(true)/\* 是否进行转换ConfirmConversions \*/,

new Variant(false)/\* 是否只读 \*/).toDispatch();

// document = Dispatch.invoke(documents, "Open", Dispatch.Method,

// new Object[] { wordFilePath, new Variant(true),

// new Variant(false)

// }, new int[1]).toDispatch();

}

// 向 document 中插入文本内容

public void insertText(String textToInsert) {

// Get the current selection within Word at the moment.

// a new document has just been created then this will be at

// the top of the new doc 获得选 中的内容，如果是一个新创建的文件，因里面无内容，则光标应处于文件开头处

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch();

// 取消选中,应该就是移动光标 ，否则 新添加的内容会覆盖选中的内容

Dispatch.call(selection, "MoveRight", new Variant(1), new Variant(1));

// Put the specified text at the insertion point

Dispatch.put(selection, "Text", textToInsert);

// 取消选中,应该就是移动光标

Dispatch.call(selection, "MoveRight", new Variant(1), new Variant(1));

}

// 向文档中添加 一个图片，

public void insertJpeg(String jpegFilePath) {

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch();

Dispatch image = Dispatch.get(selection, "InLineShapes").toDispatch();

Dispatch.call(image, "AddPicture", jpegFilePath);

}

// 段落的处理,插入格式化的文本

public void insertFormatStr(String text) {

Dispatch wordContent = Dispatch.get(document, "Content").toDispatch(); // 取得word文件的内容

Dispatch.call(wordContent, "InsertAfter", text);// 插入一个段落到最后

Dispatch paragraphs = Dispatch.get(wordContent, "Paragraphs")

.toDispatch(); // 所有段落

int paragraphCount = Dispatch.get(paragraphs, "Count").changeType(

Variant.VariantInt).getInt();// 一共的段落数

// 找到刚输入的段落，设置格式

Dispatch lastParagraph = Dispatch.call(paragraphs, "Item",

new Variant(paragraphCount)).toDispatch(); // 最后一段（也就是刚插入的）

// Range 对象表示文档中的一个连续范围，由一个起始字符位置和一个终止字符位置定义

Dispatch lastParagraphRange = Dispatch.get(lastParagraph, "Range")

.toDispatch();

Dispatch font = Dispatch.get(lastParagraphRange, "Font").toDispatch();

Dispatch.put(font, "Bold", new Variant(true)); // 设置为黑体

Dispatch.put(font, "Italic", new Variant(true)); // 设置为斜体

Dispatch.put(font, "Name", new Variant("宋体")); //

Dispatch.put(font, "Size", new Variant(12)); // 小四

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch();

Dispatch.call(selection, "TypeParagraph");// 插入一个空行

Dispatch alignment = Dispatch.get(selection, "ParagraphFormat")

.toDispatch();// 段落格式

Dispatch.put(alignment, "Alignment", "2"); // (1:置中 2:靠右 3:靠左)

}

// word 中在对表格进行遍历的时候 ，是先列后行 先column 后cell

// 另外下标从1开始

public void insertTable(String tableTitle, int row, int column) {

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象

Dispatch.call(selection, "TypeText", tableTitle); // 写入标题内容 // 标题格行

Dispatch.call(selection, "TypeParagraph"); // 空一行段落

Dispatch.call(selection, "TypeParagraph"); // 空一行段落

Dispatch.call(selection, "MoveDown"); // 游标往下一行

// 建立表格

Dispatch tables = Dispatch.get(document, "Tables").toDispatch();

// int count = Dispatch.get(tables,

// "Count").changeType(Variant.VariantInt).getInt(); // document中的表格数量

// Dispatch table = Dispatch.call(tables, "Item", new Variant(

// 1)).toDispatch();//文档中第一个表格

Dispatch range = Dispatch.get(selection, "Range").toDispatch();// /当前光标位置或者选中的区域

Dispatch newTable = Dispatch.call(tables, "Add", range,

new Variant(row), new Variant(column), new Variant(1))

.toDispatch(); // 设置row,column,表格外框宽度

Dispatch cols = Dispatch.get(newTable, "Columns").toDispatch(); // 此表的所有列，

int colCount = Dispatch.get(cols, "Count").changeType(

Variant.VariantInt).getInt();// 一共有多少列 实际上这个数==column

System.out.println(colCount + "列");

for (int i = 1; i <= colCount; i++) { // 循环取出每一列

Dispatch col = Dispatch.call(cols, "Item", new Variant(i))

.toDispatch();

Dispatch cells = Dispatch.get(col, "Cells").toDispatch();// 当前列中单元格

int cellCount = Dispatch.get(cells, "Count").changeType(

Variant.VariantInt).getInt();// 当前列中单元格数 实际上这个数等于row

for (int j = 1; j <= cellCount; j++) {// 每一列中的单元格数

// Dispatch cell = Dispatch.call(cells, "Item", new

// Variant(j)).toDispatch(); //当前单元格

// Dispatch cell = Dispatch.call(newTable, "Cell", new

// Variant(j) , new Variant(i) ).toDispatch(); //取单元格的另一种方法

// Dispatch.call(cell, "Select");//选中当前单元格

// Dispatch.put(selection, "Text",

// "第"+j+"行，第"+i+"列");//往选中的区域中填值，也就是往当前单元格填值

putTxtToCell(newTable, j, i, "第" + j + "行，第" + i + "列");// 与上面四句的作用相同

}

}

}

/\*\* \*/

/\*\*

\* 在指定的单元格里填写数据

\*

\* @param tableIndex

\* @param cellRowIdx

\* @param cellColIdx

\* @param txt

\*/

public void putTxtToCell(Dispatch table, int cellRowIdx, int cellColIdx,

String txt) {

Dispatch cell = Dispatch.call(table, "Cell", new Variant(cellRowIdx),

new Variant(cellColIdx)).toDispatch();

Dispatch.call(cell, "Select");

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象

Dispatch.put(selection, "Text", txt);

}

/\*\* \*/

/\*\*

\* 在指定的单元格里填写数据

\*

\* @param tableIndex

\* @param cellRowIdx

\* @param cellColIdx

\* @param txt

\*/

public void putTxtToCell(int tableIndex, int cellRowIdx, int cellColIdx,

String txt) {

// 所有表格

Dispatch tables = Dispatch.get(document, "Tables").toDispatch();

// 要填充的表格

Dispatch table = Dispatch.call(tables, "Item", new Variant(tableIndex))

.toDispatch();

Dispatch cell = Dispatch.call(table, "Cell", new Variant(cellRowIdx),

new Variant(cellColIdx)).toDispatch();

Dispatch.call(cell, "Select");

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象

Dispatch.put(selection, "Text", txt);

}

// 合并两个单元格

public void mergeCell(Dispatch cell1, Dispatch cell2) {

Dispatch.call(cell1, "Merge", cell2);

}

public void mergeCell(Dispatch table, int row1, int col1, int row2, int col2) {

Dispatch cell1 = Dispatch.call(table, "Cell", new Variant(row1),

new Variant(col1)).toDispatch();

Dispatch cell2 = Dispatch.call(table, "Cell", new Variant(row2),

new Variant(col2)).toDispatch();

mergeCell(cell1, cell2);

}

public void mergeCellTest() {

Dispatch tables = Dispatch.get(document, "Tables").toDispatch();

int tableCount = Dispatch.get(tables, "Count").changeType(

Variant.VariantInt).getInt(); // document中的表格数量

Dispatch table = Dispatch.call(tables, "Item", new Variant(tableCount))

.toDispatch();// 文档中最后一个table

mergeCell(table, 1, 1, 1, 2);// 将table 中x=1,y=1 与x=1,y=2的两个单元格合并

}

// ========================================================

/\*\* \*/

/\*\*

\* 把选定的内容或光标插入点向上移动

\*

\* @param pos

\* 移动的距离

\*/

public void moveUp(int pos) {

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象

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

// MoveDown MoveLeft moveRight

// moveStart ( Dispatch.call(selection, "HomeKey", new Variant(6));

// )

// moveEnd Dispatch.call(selection, "EndKey", new Variant(6));

Dispatch.call(selection, "MoveUp");

}

}

/\*\* \*/

/\*\*

\* 从选定内容或插入点开始查找文本

\*

\* @param toFindText

\* 要查找的文本

\* @return boolean true-查找到并选中该文本，false-未查找到文本

\*/

public boolean find(String toFindText) {

if (toFindText == null || toFindText.equals(""))

return false;

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象

// 从selection所在位置开始查询

Dispatch find = Dispatch.call(selection, "Find").toDispatch();

// 设置要查找的内容

Dispatch.put(find, "Text", toFindText);

// 向前查找

Dispatch.put(find, "Forward", "True");

// 设置格式

Dispatch.put(find, "Format", "True");

// 大小写匹配

Dispatch.put(find, "MatchCase", "True");

// 全字匹配

Dispatch.put(find, "MatchWholeWord", "True");

// 查找并选中

return Dispatch.call(find, "Execute").getBoolean();

}

/\*\* \*/

/\*\*

\* 把选定选定内容设定为替换文本

\*

\* @param toFindText

\* 查找字符串

\* @param newText

\* 要替换的内容

\* @return

\*/

public boolean replaceText(String toFindText, String newText) {

if (!find(toFindText))

return false;

Dispatch selection = Dispatch.get(MsWordApp, "Selection").toDispatch(); // 输入内容需要的对象

Dispatch.put(selection, "Text", newText);

return true;

}

public void printFile() {

// Just print the current document to the default printer

Dispatch.call(document, "PrintOut");

}

// 保存文档的更改

public void save() {

Dispatch.call(document, "Save");

}

public void saveFileAs(String filename) {

Dispatch.call(document, "SaveAs", filename);

}

public void closeDocument() {

// Close the document without saving changes

// 0 = wdDoNotSaveChanges

// -1 = wdSaveChanges

// -2 = wdPromptToSaveChanges

Dispatch.call(document, "Close", new Variant(0));

document = null;

}

public void closeWord() {

Dispatch.call(MsWordApp, "Quit");

MsWordApp = null;

document = null;

}

// 设置wordApp打开后窗口的位置

public void setLocation() {

Dispatch activeWindow = Dispatch.get(MsWordApp, "Application")

.toDispatch();

Dispatch.put(activeWindow, "WindowState", new Variant(1)); // 0=default

// 1=maximize

// 2=minimize

Dispatch.put(activeWindow, "Top", new Variant(0));

Dispatch.put(activeWindow, "Left", new Variant(0));

Dispatch.put(activeWindow, "Height", new Variant(600));

Dispatch.put(activeWindow, "width", new Variant(800));

}

}

public class JacobTest2 {

public static void createANewFileTest() {

WordBean wordBean = new WordBean();

// word.openWord(true);// 打开 word 程序

wordBean.setVisible(true);

wordBean.createNewDocument();// 创建一个新文档

wordBean.setLocation();// 设置打开后窗口的位置

wordBean.insertText("你好");// 向文档中插入字符

wordBean.insertJpeg("D:" + File.separator + "a.jpg"); // 插入图片

// 如果 ，想保存文件，下面三句

// word.saveFileAs("d://a.doc");

// word.closeDocument();

// word.closeWord();

}

public static void openAnExistsFileTest() {

WordBean wordBean = new WordBean();

wordBean.setVisible(true); // 是否前台打开word 程序，或者后台运行

wordBean.openFile("d://a.doc");

wordBean.insertJpeg("D:" + File.separator + "a.jpg"); // 插入图片(注意刚打开的word

// ，光标处于开头，故，图片在最前方插入)

wordBean.save();

wordBean.closeDocument();

wordBean.closeWord();

}

public static void insertFormatStr(String str) {

WordBean wordBean = new WordBean();

wordBean.setVisible(true); // 是否前台打开word 程序，或者后台运行

wordBean.createNewDocument();// 创建一个新文档

wordBean.insertFormatStr(str);// 插入一个段落，对其中的字体进行了设置

}

public static void insertTableTest() {

WordBean wordBean = new WordBean();

wordBean.setVisible(true); // 是否前台打开word 程序，或者后台运行

wordBean.createNewDocument();// 创建一个新文档

wordBean.setLocation();

wordBean.insertTable("表名", 3, 2);

wordBean.saveFileAs("d://table.doc");

wordBean.closeDocument();

wordBean.closeWord();

}

public static void mergeTableCellTest() {

insertTableTest();//生成d://table.doc

WordBean wordBean = new WordBean();

wordBean.setVisible(true); // 是否前台打开word 程序，或者后台运行

wordBean.openFile("d://table.doc");

wordBean.mergeCellTest();

}

public static void main(String[] args) {

// 进行测试前要保证d://a.jpg 图片文件存在

// createANewFileTest();//创建一个新文件

// openAnExistsFileTest();// 打开一个存在 的文件

// insertFormatStr("格式 化字符串");//对字符串进行一定的修饰

//insertTableTest();// 创建一个表格

mergeTableCellTest();// 对表格中的单元格进行合并

}

}

# 第三部分

java写入word表格模板(欢迎高手过来探讨)

我有一份word表格模板,格式是规定的了

数据是从数据库读出来的,现在想把读出来的数据插入到模板的相应位置

大家有什么好建议吗??

我知道用jacod组件可以,但是jacod效率太低,大家如果有其他好的方法,麻烦提供一下,谢谢!

**问题补充：**

BarryWei有没有Demo呢??有的话可以发给我吗??

falcon1990是利用poi对excel表格操作,这个我也会啊,我要的是对word的表格

**问题补充：**

java 下的 **该问题已经关闭:** 既然大家都没有什么好点的方法,我只能去操作2007的XML啦,推荐一篇文章给大家 http://www.infoq.com/cn/articles/cracking-office-2007-with-java

回答

首先，在word模板中适当的位置处加上特殊标记，比如 $内容$。

接着，用BufferedReader把模板中的所有字符串取出来，放入StringBuffer对象中。

第三，将StringBuffer对象中所有$内容$ replace掉，换成数据库中取出的结果。

第四，done。

package cn.com.enjoysoft.connection.db;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.OutputStream;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.poi.hssf.usermodel.HSSFCell;

import org.apache.poi.hssf.usermodel.HSSFCellStyle;

import org.apache.poi.hssf.usermodel.HSSFFont;

import org.apache.poi.hssf.usermodel.HSSFRichTextString;

import org.apache.poi.hssf.usermodel.HSSFRow;

import org.apache.poi.hssf.usermodel.HSSFSheet;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.hssf.util.HSSFColor;

@SuppressWarnings("serial")

public class DownExcel extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

String fileName = sdf.format(new Date()) + "导出数据";

fileName = response.encodeURL(new String(fileName.getBytes("GBK"), "iso8859-1"));// 转码

response.setContentType("application/octet-stream");

response.setHeader("Content-Disposition", "attachement;filename="

+ fileName + ".xls");

String type = request.getParameter("type");

try {

OutputStream os = response.getOutputStream();

HSSFWorkbook wb = generateWorkbook(type);

if (wb != null) {

wb.write(os);

}

os.flush();

os.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

static void processCell(HSSFCell cell, int value, HSSFCellStyle style) {

cell.setCellValue(value);

cell.setCellStyle(style);

}

static void processCell(HSSFCell cell, String value, HSSFCellStyle style) {

HSSFRichTextString \_value = new HSSFRichTextString(value);

cell.setCellValue(\_value);

cell.setCellStyle(style);

}

private HSSFWorkbook generateWorkbook(String type) {

Connection conn = null;

try {

conn = TodayConnection.getConnection();

Statement stmt = conn.createStatement();

String sql = null;

if (type == null || type.equals("")) {

sql = "select \* from telNumber order by id desc";

} else {

sql = "select \* from telNumber where counseling ='" + type + "' order by id desc";

}

ResultSet rs = stmt.executeQuery(sql);

HSSFWorkbook workbook = new HSSFWorkbook();

HSSFSheet sheet = workbook.createSheet("导出数据");

HSSFRow row = sheet.createRow((short) 0);

row.setHeight((short) (15 \* 32));

HSSFCell[] headArray = new HSSFCell[7];

String[] headTitles = new String[] { "No.", "姓名", "电话", "公司名称",

"咨询项目", "备注", "时间" };

HSSFCellStyle style = workbook.createCellStyle();

// 这里解决背景颜色

style.setFillForegroundColor(HSSFColor.GREY\_25\_PERCENT.index);

style.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND);

style.setBorderBottom(HSSFCellStyle.BORDER\_THIN);

style.setBottomBorderColor(HSSFColor.BLACK.index);

style.setBorderLeft(HSSFCellStyle.BORDER\_THIN);

style.setLeftBorderColor(HSSFColor.BLACK.index);

style.setBorderRight(HSSFCellStyle.BORDER\_THIN);

style.setRightBorderColor(HSSFColor.BLACK.index);

style.setBorderTop(HSSFCellStyle.BORDER\_THIN);

style.setTopBorderColor(HSSFColor.BLACK.index);

style.setAlignment(HSSFCellStyle.ALIGN\_CENTER);

// 字体

HSSFFont font = workbook.createFont();

font.setFontHeightInPoints((short) 12);

font.setFontName("Courier New");

font.setBoldweight(HSSFFont.BOLDWEIGHT\_BOLD);

style.setFont(font);

for (int i = 0; i < headArray.length; i++) {

headArray[i] = row.createCell(i);

headArray[i].setCellStyle(style);

HSSFRichTextString value = new HSSFRichTextString(headTitles[i]);

headArray[i].setCellValue(value);

}

HSSFCellStyle styleData = workbook.createCellStyle();

styleData.setBorderBottom(HSSFCellStyle.BORDER\_THIN);

styleData.setBottomBorderColor(HSSFColor.BLACK.index);

styleData.setBorderLeft(HSSFCellStyle.BORDER\_THIN);

styleData.setLeftBorderColor(HSSFColor.BLACK.index);

styleData.setBorderRight(HSSFCellStyle.BORDER\_THIN);

styleData.setRightBorderColor(HSSFColor.BLACK.index);

styleData.setBorderTop(HSSFCellStyle.BORDER\_THIN);

styleData.setTopBorderColor(HSSFColor.BLACK.index);

styleData.setAlignment(HSSFCellStyle.ALIGN\_CENTER);

// 字体

HSSFFont font1 = workbook.createFont();

font1.setFontHeightInPoints((short) 12);

font1.setFontName("Courier New");

styleData.setFont(font1);

// 循环查询的所有记录把每一行写入worksheet里面

int no = 1;

short rowNo = 1;

while (rs.next()) {

HSSFRow dataRow = sheet.createRow(rowNo++);

dataRow.setHeight((short) (12 \* 32));

processCell(dataRow.createCell(0), no++, styleData);

processCell(dataRow.createCell(1), rs.getString("name"),

styleData);

processCell(dataRow.createCell(2), rs.getString("telNumber"),

styleData);

processCell(dataRow.createCell(3), rs.getString("corp"),

styleData);

processCell(dataRow.createCell(4), rs.getString("counseling"),

styleData);

processCell(dataRow.createCell(5), rs.getString("remark"),

styleData);

processCell(dataRow.createCell(6), rs.getString("jointime"),

styleData);

}

// 调整宽度

short[] widths = new short[] { 5, 15, 18, 22, 13, 50, 30 };

for (short k = 0; k < 7; k++) {

sheet.setColumnWidth(k, widths[k] \* 256);

}

return workbook;

} catch (Exception e) {

} finally {

try {

// 关闭数据库连接

if (conn != null && !conn.isClosed())

conn.close();

} catch (SQLException e) {

}

}// END OF PROGRAMM

return null;

}

}

这里是详细的代码！需要poi-3.2-FINAL-20081019.jar自己下就可以了！

如果是jsp页面的话，那么生成word很简单的，我以前做的那个系统最后就有个报表要提供下载和打印，都要求是word文档，我的解决方法：

1、做好一个table布局的jsp页面，在里面添好表名、表头，样式等

2、在servlet或者action中调用biz读取数据，然后写入session或者request中

3、转向到jsp页面，在页面中读取session或者request中的数据

4、要注意的是，这个jsp页面不是一般的页面，这个jsp页面的content-type要设置，改成这样：

<meta http-equiv="Content-Type" content="application/msword; charset=gb2312" />

这样就可以提供word格式的文档，而且会在客户端提供下载。

你要客户直接把做好的word下载下来就ok，稍微编辑一些字体大小或者样式（或者这些东西在jsp页面中全部做好），直接打印或者备案就ok。

希望对你有用。