**说明**

  POI可以对2003-和2007+版本的Excel文件做导入导出操作，本章只简单介绍对Excel文件的导入操作。

       Excel文件的上传处理处理请求，依然使用SpringMvc中的MultipartRequest方式处理。

       前端JSP中使用传统form表单提交方式和Juery.form.js插件提供的异步表单请求方式，分别对这两种方式进行介绍。

**环境**

JDK7+ Tomcat7.x + Spring4.1.8

**说明**：

ImportExcelUtil.[**Java**](http://lib.csdn.net/base/17)：Excel解析工具类

UploadExcelControl.java ：处理来自页面的请求控制器

InfoVo.java ：将Excel转换为对象存储

main.jsp：前端访问页

........

**ImportExcelUtil.java（Excel解析工具类）**

**[java]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

1. **package** com.poiexcel.util;
3. **import** java.io.IOException;
4. **import** java.io.InputStream;
5. **import** java.text.DecimalFormat;
6. **import** java.text.SimpleDateFormat;
7. **import** java.util.ArrayList;
8. **import** java.util.List;
10. **import** org.apache.poi.hssf.usermodel.HSSFWorkbook;
11. **import** org.apache.poi.ss.usermodel.Cell;
12. **import** org.apache.poi.ss.usermodel.Row;
13. **import** org.apache.poi.ss.usermodel.Sheet;
14. **import** org.apache.poi.ss.usermodel.Workbook;
15. **import** org.apache.poi.xssf.usermodel.XSSFWorkbook;

18. **public** **class** ImportExcelUtil {
20. **private** **final** **static** String excel2003L =".xls";    //2003- 版本的excel
21. **private** **final** **static** String excel2007U =".xlsx";   //2007+ 版本的excel
23. /\*\*
24. \* 描述：获取IO流中的数据，组装成List<List<Object>>对象
25. \* @param in,fileName
26. \* @return
27. \* @throws IOException
28. \*/
29. **public**  List<List<Object>> getBankListByExcel(InputStream in,String fileName) **throws** Exception{
30. List<List<Object>> list = **null**;
32. //创建Excel工作薄
33. Workbook work = **this**.getWorkbook(in,fileName);
34. **if**(**null** == work){
35. **throw** **new** Exception("创建Excel工作薄为空！");
36. }
37. Sheet sheet = **null**;
38. Row row = **null**;
39. Cell cell = **null**;
41. list = **new** ArrayList<List<Object>>();
42. //遍历Excel中所有的sheet
43. **for** (**int** i = 0; i < work.getNumberOfSheets(); i++) {
44. sheet = work.getSheetAt(i);
45. **if**(sheet==**null**){**continue**;}
47. //遍历当前sheet中的所有行
48. **for** (**int** j = sheet.getFirstRowNum(); j < sheet.getLastRowNum(); j++) {
49. row = sheet.getRow(j);
50. **if**(row==**null**||row.getFirstCellNum()==j){**continue**;}
52. //遍历所有的列
53. List<Object> li = **new** ArrayList<Object>();
54. **for** (**int** y = row.getFirstCellNum(); y < row.getLastCellNum(); y++) {
55. cell = row.getCell(y);
56. li.add(**this**.getCellValue(cell));
57. }
58. list.add(li);
59. }
60. }
61. work.close();
62. **return** list;
63. }
65. /\*\*
66. \* 描述：根据文件后缀，自适应上传文件的版本
67. \* @param inStr,fileName
68. \* @return
69. \* @throws Exception
70. \*/
71. **public**  Workbook getWorkbook(InputStream inStr,String fileName) **throws** Exception{
72. Workbook wb = **null**;
73. String fileType = fileName.substring(fileName.lastIndexOf("."));
74. **if**(excel2003L.equals(fileType)){
75. wb = **new** HSSFWorkbook(inStr);  //2003-
76. }**else** **if**(excel2007U.equals(fileType)){
77. wb = **new** XSSFWorkbook(inStr);  //2007+
78. }**else**{
79. **throw** **new** Exception("解析的文件格式有误！");
80. }
81. **return** wb;
82. }
84. /\*\*
85. \* 描述：对表格中数值进行格式化
86. \* @param cell
87. \* @return
88. \*/
89. **public**  Object getCellValue(Cell cell){
90. Object value = **null**;
91. DecimalFormat df = **new** DecimalFormat("0");  //格式化number String字符
92. SimpleDateFormat sdf = **new** SimpleDateFormat("yyy-MM-dd");  //日期格式化
93. DecimalFormat df2 = **new** DecimalFormat("0.00");  //格式化数字
95. **switch** (cell.getCellType()) {
96. **case** Cell.CELL\_TYPE\_STRING:
97. value = cell.getRichStringCellValue().getString();
98. **break**;
99. **case** Cell.CELL\_TYPE\_NUMERIC:
100. **if**("General".equals(cell.getCellStyle().getDataFormatString())){
101. value = df.format(cell.getNumericCellValue());
102. }**else** **if**("m/d/yy".equals(cell.getCellStyle().getDataFormatString())){
103. value = sdf.format(cell.getDateCellValue());
104. }**else**{
105. value = df2.format(cell.getNumericCellValue());
106. }
107. **break**;
108. **case** Cell.CELL\_TYPE\_BOOLEAN:
109. value = cell.getBooleanCellValue();
110. **break**;
111. **case** Cell.CELL\_TYPE\_BLANK:
112. value = "";
113. **break**;
114. **default**:
115. **break**;
116. }
117. **return** value;
118. }

121. }

**UploadExcelControl.java (**[**spring**](http://lib.csdn.net/base/17)**控制器)**

**[java]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

1. **package** com.poiexcel.control;
3. **import** java.io.InputStream;
4. **import** java.io.PrintWriter;
5. **import** java.util.List;
7. **import** javax.servlet.http.HttpServletRequest;
8. **import** javax.servlet.http.HttpServletResponse;
10. **import** org.springframework.stereotype.Controller;
11. **import** org.springframework.web.bind.annotation.RequestMapping;
12. **import** org.springframework.web.bind.annotation.RequestMethod;
13. **import** org.springframework.web.bind.annotation.ResponseBody;
14. **import** org.springframework.web.multipart.MultipartFile;
15. **import** org.springframework.web.multipart.MultipartHttpServletRequest;
17. **import** com.poiexcel.util.ImportExcelUtil;
18. **import** com.poiexcel.vo.InfoVo;
20. @Controller
21. @RequestMapping("/uploadExcel/\*")
22. **public** **class** UploadExcelControl {
24. /\*\*
25. \* 描述：通过传统方式form表单提交方式导入excel文件
26. \* @param request
27. \* @throws Exception
28. \*/
29. @RequestMapping(value="upload.do",method={RequestMethod.GET,RequestMethod.POST})
30. **public**  String  uploadExcel(HttpServletRequest request) **throws** Exception {
31. MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;
32. System.out.println("通过传统方式form表单提交方式导入excel文件！");
34. InputStream in =**null**;
35. List<List<Object>> listob = **null**;
36. MultipartFile file = multipartRequest.getFile("upfile");
37. **if**(file.isEmpty()){
38. **throw** **new** Exception("文件不存在！");
39. }
40. in = file.getInputStream();
41. listob = **new** ImportExcelUtil().getBankListByExcel(in,file.getOriginalFilename());
42. in.close();
44. //该处可调用service相应方法进行数据保存到数据库中，现只对数据输出
45. **for** (**int** i = 0; i < listob.size(); i++) {
46. List<Object> lo = listob.get(i);
47. InfoVo vo = **new** InfoVo();
48. vo.setCode(String.valueOf(lo.get(0)));
49. vo.setName(String.valueOf(lo.get(1)));
50. vo.setDate(String.valueOf(lo.get(2)));
51. vo.setMoney(String.valueOf(lo.get(3)));
53. System.out.println("打印信息-->机构:"+vo.getCode()+"  名称："+vo.getName()+"   时间："+vo.getDate()+"   资产："+vo.getMoney());
54. }
55. **return** "result";
56. }
58. /\*\*
59. \* 描述：通过 jquery.form.js 插件提供的ajax方式上传文件
60. \* @param request
61. \* @param response
62. \* @throws Exception
63. \*/
64. @ResponseBody
65. @RequestMapping(value="ajaxUpload.do",method={RequestMethod.GET,RequestMethod.POST})
66. **public**  **void**  ajaxUploadExcel(HttpServletRequest request,HttpServletResponse response) **throws** Exception {
67. MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest) request;
69. System.out.println("通过 jquery.form.js 提供的ajax方式上传文件！");
71. InputStream in =**null**;
72. List<List<Object>> listob = **null**;
73. MultipartFile file = multipartRequest.getFile("upfile");
74. **if**(file.isEmpty()){
75. **throw** **new** Exception("文件不存在！");
76. }
78. in = file.getInputStream();
79. listob = **new** ImportExcelUtil().getBankListByExcel(in,file.getOriginalFilename());
81. //该处可调用service相应方法进行数据保存到数据库中，现只对数据输出
82. **for** (**int** i = 0; i < listob.size(); i++) {
83. List<Object> lo = listob.get(i);
84. InfoVo vo = **new** InfoVo();
85. vo.setCode(String.valueOf(lo.get(0)));
86. vo.setName(String.valueOf(lo.get(1)));
87. vo.setDate(String.valueOf(lo.get(2)));
88. vo.setMoney(String.valueOf(lo.get(3)));
90. System.out.println("打印信息-->机构:"+vo.getCode()+"  名称："+vo.getName()+"   时间："+vo.getDate()+"   资产："+vo.getMoney());
91. }
93. PrintWriter out = **null**;
94. response.setCharacterEncoding("utf-8");  //防止ajax接受到的中文信息乱码
95. out = response.getWriter();
96. out.print("文件导入成功！");
97. out.flush();
98. out.close();
99. }

102. }

**InfoVo.java(保存Excel数据对应的对象)**

**[java]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

1. **package** com.poiexcel.vo;

4. //将Excel每一行数值转换为对象
5. **public** **class** InfoVo {
7. **private** String code;
8. **private** String name;
9. **private** String date;
10. **private** String money;
12. **public** String getCode() {
13. **return** code;
14. }
15. **public** **void** setCode(String code) {
16. **this**.code = code;
17. }
18. **public** String getName() {
19. **return** name;
20. }
21. **public** **void** setName(String name) {
22. **this**.name = name;
23. }
24. **public** String getDate() {
25. **return** date;
26. }
27. **public** **void** setDate(String date) {
28. **this**.date = date;
29. }
30. **public** String getMoney() {
31. **return** money;
32. }
33. **public** **void** setMoney(String money) {
34. **this**.money = money;
35. }
36. }

**main.jsp(前端代码)**

**[html]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

1. **<**%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%**>**
2. **<**%
3. String path = request.getContextPath();
4. String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";
5. %**>**
6. **<html>**
7. **<head>**
8. **<base** href="<%=basePath%>"**>**
9. **<script** type="text/javascript" src="js/jquery-2.1.4.min.js"**></script>**
10. **<script** type="text/javascript" src="js/jquery.form.js"**></script>**
11. **<title>**My JSP 'index.jsp' starting page**</title>**
12. **<script** type="text/javascript"**>**
13. //ajax 方式上传文件操作
14. $(document).ready(function(){
15. $('#btn').click(function(){
16. if(checkData()){
17. $('#form1').ajaxSubmit({
18. url:'uploadExcel/ajaxUpload.do',
19. dataType: 'text',
20. success: resutlMsg,
21. error: errorMsg
22. });
23. function resutlMsg(msg){
24. alert(msg);
25. $("#upfile").val("");
26. }
27. function errorMsg(){
28. alert("导入excel出错！");
29. }
30. }
31. });
32. });
34. //JS校验form表单信息
35. function checkData(){
36. var fileDir = $("#upfile").val();
37. var suffix = fileDir.substr(fileDir.lastIndexOf("."));
38. if("" == fileDir){
39. alert("选择需要导入的Excel文件！");
40. return false;
41. }
42. if(".xls" != suffix && ".xlsx" != suffix ){
43. alert("选择Excel格式的文件导入！");
44. return false;
45. }
46. return true;
47. }
48. **</script>**
49. **</head>**
51. **<body>**
52. **<div>**1.通过简单的form表单提交方式，进行文件的上**</br>** 2.通过jquery.form.js插件提供的form表单一步提交功能 **</div></br>**
53. **<form** method="POST"  enctype="multipart/form-data" id="form1" action="uploadExcel/upload.do"**>**
54. **<table>**
55. **<tr>**
56. **<td>**上传文件: **</td>**
57. **<td>** **<input** id="upfile" type="file" name="upfile"**></td>**
58. **</tr>**
59. **<tr>**
60. **<td><input** type="submit" value="提交" onclick="return checkData()"**></td>**
61. **<td><input** type="button" value="ajax方式提交" id="btn" name="btn" **></td>**
62. **</tr>**
63. **</table>**
64. **</form>**
66. **</body>**
67. **</html>**

**web.xml**

**[html]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="3.0"
3. xmlns="http://java.sun.com/xml/ns/javaee"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6. http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd "**>**
8. <!-- 加载Spring容器监听 -->
9. **<listener>**
10. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
11. **</listener>**
13. <!-- 设置Spring容器加载配置文件路径 -->
14. **<context-param>**
15. **<param-name>**contextConfigLocation**</param-name>**
16. **<param-value>**WEB-INF/application/applicationContext-\*.xml**</param-value>**
17. **</context-param>**
19. <!--配置Springmvc核心控制器-->
20. **<servlet>**
21. **<servlet-name>**springmvc**</servlet-name>**
22. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
23. **<load-on-startup>**1**</load-on-startup>**
24. **</servlet>**
25. <!--为DispatcherServlet建立映射 -->
26. **<servlet-mapping>**
27. **<servlet-name>**springmvc**</servlet-name>**
28. **<url-pattern>**\*.do**</url-pattern>**
29. **</servlet-mapping>**
30. **<welcome-file-list>**
31. **<welcome-file>**main.jsp**</welcome-file>**
32. **</welcome-file-list>**
33. **</web-app>**

**springmvc-servlet.xml（只做简单配置）**

**[html]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

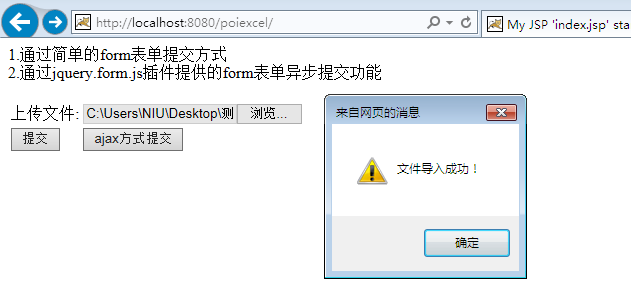
1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:context="http://www.springframework.org/schema/context"
4. xmlns:mvc="http://www.springframework.org/schema/mvc"
5. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6. xmlns:tx="http://www.springframework.org/schema/tx"
7. xsi:schemaLocation="
8. http://www.springframework.org/schema/beans
9. http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
10. http://www.springframework.org/schema/context
11. http://www.springframework.org/schema/context/spring-context-4.1.xsd
12. http://www.springframework.org/schema/mvc
13. http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd
14. http://www.springframework.org/schema/tx
15. http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
16. http://www.springframework.org/schema/context
17. http://www.springframework.org/schema/beans/spring-beans-4.1.xsd "**>**
19. <!-- 启动注解驱动-->
20. **<mvc:annotation-driven/>**
21. <!-- 启动包扫描功能-->
22. **<context:component-scan** base-package="com.poiexcel.\*" **/>**
24. **</beans>**

applicationContext-base.xml

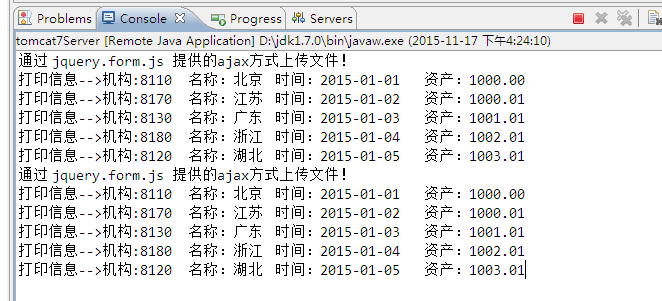
**[html]** [view plain](http://blog.csdn.net/onepersontz/article/details/49891405) [copy](http://blog.csdn.net/onepersontz/article/details/49891405)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:context="http://www.springframework.org/schema/context"
4. xmlns:mvc="http://www.springframework.org/schema/mvc"
5. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6. xmlns:tx="http://www.springframework.org/schema/tx"
7. xsi:schemaLocation="
8. http://www.springframework.org/schema/beans
9. http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
10. http://www.springframework.org/schema/context
11. http://www.springframework.org/schema/context/spring-context-4.1.xsd
12. http://www.springframework.org/schema/mvc
13. http://www.springframework.org/schema/mvc/spring-mvc-4.1.xsd
14. http://www.springframework.org/schema/tx
15. http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
16. http://www.springframework.org/schema/context
17. http://www.springframework.org/schema/context/spring-context-4.1.xsd "**>**
19. <!-- 视图解释类 -->
20. **<bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
21. **<property** name="prefix" value="/WEB-INF/jsp/" **/>**
22. **<property** name="suffix" value=".jsp" **/>**
23. **</bean>**
25. <!-- SpringMVC上传文件时，需要配置MultipartResolver处理器 -->
26. **<bean** id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"**>**
27. **<property** name="defaultEncoding" value="utf-8" **/>**
28. <!-- 指定所上传文件的总大小不能超过10485760000B。注意maxUploadSize属性的限制不是针对单个文件，而是所有文件的容量之和 -->
29. **<property** name="maxUploadSize" value="10485760000" **/>**
30. **<property** name="maxInMemorySize" value="40960" **/>**
31. **</bean>**
33. **</beans>**

**效果**

****

**后台打印信息**

****