《POI 操作 Excel》

1.写入文件

1.1 写入 Excel 97 .xls

用到 jar 包：

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>3.17</version>
</dependency>
```

测试代码

```
public static void writeExcelXLS() {
    // 第一步，创建一个 webbook，对应一个 Excel 文件
    HSSFWorkbook wb = new HSSFWorkbook();
    // 第二步，在 webbook 中添加一个 sheet,对应 Excel 文件中的 sheet
    HSSFSheet sheet = wb.createSheet("学生表一");
    // 第三步,在 sheet 中添加表头第 0 行,注意老版本 poi 对 Excel 的行数列数有限制 short
    HSSFRow row = sheet.createRow(0);
    // 第四步，创建单元格，并设置值表头 设置表头居中
    HSSFCellStyle style = wb.createCellStyle();
    style.setAlignment(HorizontalAlignment.CENTER); // 创建一个居中格式
    //创建单元格，row 是前面创建的第 0 行，所以这个是 0 行 0 列
    HSSFCell cell01 = row.createCell(0);
    cell01.setCellValue("学号");
    cell01.setCellStyle(style);
    HSSFCell cell02 = row.createCell(1);//0 行 1 列
    cell02.setCellValue("姓名");
    cell02.setCellStyle(style);
    HSSFCell cell03 = row.createCell(2);//0 行 2 列
    cell03.setCellValue("年龄");
    cell03.setCellStyle(style);
    HSSFCell cell04 = row.createCell(3);//0 行 3 列
    cell04.setCellValue("生日");
    cell04.setCellStyle(style);

    // 第五步，写入实体数据 实际应用中这些数据从数据库得到，
    //第 2 行(第一行是表头)
    HSSFRow row1 = sheet.createRow(1);
    // 创建单元格，并设置值
    row1.createCell(0).setCellValue(1);
    row1.createCell(1).setCellValue("张三");
    row1.createCell(2).setCellValue(15);
```

```
    row1.createCell(3).setCellValue(new SimpleDateFormat("yyyy-MM-dd").format(new
Date()));
    //创建新行第三行
    row1 = sheet.createRow(2);
    // 创建单元格，并设置值
    row1.createCell(0).setCellValue(2);
    row1.createCell(1).setCellValue("李四");
    row1.createCell(2).setCellValue(16);
    row1.createCell(3).setCellValue(new SimpleDateFormat("yyyy-MM-dd").format(new
Date()));

    // 第六步，将文件存到指定位置
    try {
        FileOutputStream fout = new FileOutputStream("E:/students.xls");
        wb.write(fout);
        wb.close();
        fout.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

1.2 写入 Excel 07 .xlsx

用到的 jar 包

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>3.17</version>
</dependency>
```

测试代码：

```
public static void writeExcelXLSX() {
    // 第一步，创建一个 webbook，对应一个 Excel 文件
    SXSSFWorkbook workbook = new SXSSFWorkbook();
    // 第二步，在 webbook 中添加一个 sheet,对应 Excel 文件中的 sheet
    SXSSFSheet sheet = workbook.createSheet("学生表一");
    // 第三步,在 sheet 中添加表头第 0 行,注意老版本 poi 对 Excel 的行数列数有限制 short
    SXSSFRow row = sheet.createRow(0);
    // 第四步，创建单元格，并设置值表头 设置表头居中
    CellStyle style = workbook.createCellStyle();
    style.setAlignment(HorizontalAlignment.CENTER); // 创建一个居中格式
    //创建单元格，row 是前面创建的第 0 行，所以这个是 0 行 0 列
    SXSSFCell cell01 = row.createCell(0);
    cell01.setCellValue("学号");
    cell01.setCellStyle(style);
    SXSSFCell cell02 = row.createCell(1);//0 行 1 列
    cell02.setCellValue("姓名");
    cell02.setCellStyle(style);
    SXSSFCell cell03 = row.createCell(2);//0 行 2 列
    cell03.setCellValue("年龄");
```

```
    cell03.setCellStyle(style);
    SXSSFCell cell04 = row.createCell(3);//0 行 3 列
    cell04.setCellValue("生日");
    cell04.setCellStyle(style);

    // 第五步，写入实体数据 实际应用中这些数据从数据库得到，
    //第 1 行
    SXSSFRow row1 = sheet.createRow(1);
    // 创建单元格，并设置值
    row1.createCell(0).setCellValue(1);
    row1.createCell(1).setCellValue("张三");
    row1.createCell(2).setCellValue(15);
    row1.createCell(3).setCellValue(new SimpleDateFormat("yyyy-mm-dd").format(new
Date()));
    //创建新行
    row1 = sheet.createRow(2);
    // 创建单元格，并设置值
    row1.createCell(0).setCellValue(2);
    row1.createCell(1).setCellValue("李四");
    row1.createCell(2).setCellValue(16);
    row1.createCell(3).setCellValue(new SimpleDateFormat("yyyy-MM-dd").format(new
Date()));

    // 第六步，将文件存到指定位置
    try {
        FileOutputStream fout = new FileOutputStream("E:/students.xlsx");
        workbook.write(fout);
        workbook.close();
        fout.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

1.3 写入 Excel 兼容 xls 和 xlsx

```
public static File writeExcelAll(String excelExtName) throws Exception {
    // 第一步，创建一个 webbook，对应一个 Excel 文件
    Workbook workbook = null;
    if ("xls".equals(excelExtName)) {
        workbook = new HSSFWorkbook();
    } else if ("xlsx".equals(excelExtName)) {
        workbook = new SXSSFWorkbook();
    } else {
        throw new Exception("当前文件不是 excel 文件");
    }
    // 第二步，在 webbook 中添加一个 sheet,对应 Excel 文件中的 sheet
    Sheet sheet = workbook.createSheet("学生表一");
    // 第三步,在 sheet 中添加表头第 0 行,注意老版本 poi 对 Excel 的行数列数有限制 short
    Row row = sheet.createRow(0);
```

```
    // 第四步，创建单元格，并设置值表头 设置表头居中
    CellStyle style = workbook.createCellStyle();
    style.setAlignment(HorizontalAlignment.CENTER); // 创建一个居中格式
    //创建单元格，row 是前面创建的第 0 行，所以这个是 0 行 0 列
    Cell cell01 = row.createCell(0);
    cell01.setCellValue("学号");
    cell01.setCellStyle(style);
    Cell cell02 = row.createCell(1);//0 行 1 列
    cell02.setCellValue("姓名");
    cell02.setCellStyle(style);
    Cell cell03 = row.createCell(2);//0 行 2 列
    cell03.setCellValue("年龄");
    cell03.setCellStyle(style);
    Cell cell04 = row.createCell(3);//0 行 3 列
    cell04.setCellValue("生日");
    cell04.setCellStyle(style);

    // 第五步，写入实体数据 实际应用中这些数据从数据库得到，
    //第 1 行
    Row row1 = sheet.createRow(1);
    // 创建单元格，并设置值
    row1.createCell(0).setCellValue(1);
    row1.createCell(1).setCellValue("张三");
    row1.createCell(2).setCellValue(15);
    row1.createCell(3).setCellValue(new SimpleDateFormat("yyyy-mm-dd").format(new
Date()));
    //创建新行
    row1 = sheet.createRow(2);
    // 创建单元格，并设置值
    row1.createCell(0).setCellValue(2);
    row1.createCell(1).setCellValue("李四");
    row1.createCell(2).setCellValue(16);
    row1.createCell(3).setCellValue(new SimpleDateFormat("yyyy-mm-dd").format(new
Date()));

    // 第六步，将文件存到指定位置
    try {
        File excel = new File("student1." + excelExtName);
        FileOutputStream fout = new FileOutputStream(excel);
        workbook.write(fout);
        workbook.close();
        fout.close();
        return excel;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

2.读取 Excel

```
public static List<List<String>> readExcelXLS(File file) throws Exception {
    FileInputStream is = new FileInputStream(file);
    Workbook workbook = WorkbookFactory.create(is);
    int sheetCount = workbook.getNumberOfSheets();  //Sheet 的数量
    //这里可以遍历每个 sheet 略
    Sheet sheet = workbook.getSheetAt(0);//第一个 sheet
    int rowCount = sheet.getPhysicalNumberOfRows();//总行数
    List<List<String>> sheetContent = new ArrayList<>();
    for (int r = 0; r < rowCount; r++) {
        List<String> rowContent = new ArrayList<>();
        Row row = sheet.getRow(r);
        int cellCount = row.getPhysicalNumberOfCells(); //获取总列数
        for (int c = 0; c < cellCount; c++) {
            Cell cell = row.getCell(c);
            CellType cellType = cell.getCellTypeEnum();
            String cellValue = null;
            switch (cellType) {
                case STRING: //文本
                    cellValue = cell.getStringCellValue();
                    break;
                case NUMERIC: //数字、日期
                    if (DateUtil.isCellDateFormatted(cell)) {
                        SimpleDateFormat fmt = new SimpleDateFormat("yyyy-MM-dd");
                        cellValue = fmt.format(cell.getDateCellValue()); //日期型
                    } else {
                        cellValue = String.valueOf(cell.getNumericCellValue()); //
数字
                    }
                    break;
                case BOOLEAN: //布尔型
                    cellValue = String.valueOf(cell.getBooleanCellValue());
                    break;
                case _NONE: //空白
                    cellValue = cell.getStringCellValue();
                    break;
                case ERROR: //错误
                    cellValue = "错误";
                    break;
                case FORMULA: //公式
                    cellValue = "错误";
                    break;
                default:
                    cellValue = "错误";
            }
            System.out.print(cellValue + "    ");
            rowContent.add(cellValue);
        }
```

```
            sheetContent.add(rowContent);
            System.out.println();
        }
        workbook.close();
        is.close();
        return sheetContent;
    }
```

3.上传 Excel 并读取数据

web 环境需要的 jar：

```
<!--spring 相关包-->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>4.3.11.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>4.3.11.RELEASE</version>
</dependency>

<!-- 文件上传所依赖的 jar 包 -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>

<!--Servlet 相关包-->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
</dependency>
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

springMVC-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
```

```
  http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
  http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.2.xsd
  http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd">
    <!-- 启动注解，注册服务-->
    <mvc:annotation-driven/>

    <!-- 启动自动扫描 -->
    <context:component-scan base-package="controller">
        <!-- 制定扫包规则 ,只扫描使用@Controller 注解的 JAVA 类 -->
        <context:include-filter type="annotation"
expression="org.springframework.stereotype.Controller"/>
    </context:component-scan>

    <!--1*1024*1024 即 1M resolveLazily 属性启用是为了推迟文件解析，以便捕获文件大小
异常 -->
    <!--文件上传解析器-->
    <bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
        <property name="maxUploadSize" value="1048576"/>
        <property name="defaultEncoding" value="UTF-8"/>
        <property name="resolveLazily" value="true"/>
    </bean>

</beans>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
          http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
  <listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-cla
ss>
  </listener>

  <servlet>
    <servlet-name>springMVC</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>springMVC</servlet-name>
    <url-pattern>*.html</url-pattern>
  </servlet-mapping>

</web-app>
```

Controller

```
@RequestMapping("/upload.html")
public ModelAndView upload(@RequestParam("file") MultipartFile file) throws Exception
{
    File excel = new File(file.getOriginalFilename());
    file.transferTo(excel);
    List<List<String>> data = PoiExcelUtil.readExcelXLS(excel);
    return new ModelAndView("/index.jsp", "data", data);
}
```

页面：

```
<form action="/upload.html" enctype="multipart/form-data" method="post">
    <input name="file" type="file"/><input type="submit"/>
</form>
<c:forEach items="${data}" var="str">
    <tr>
        <c:forEach items="${str}" var="s">
            <td>${s}</td>
        </c:forEach>
    </tr>
</c:forEach>
```

4.导出并下载 Excel

Controller

```
@RequestMapping("/download.html")
public void download(HttpServletResponse response) throws Exception {
    File excel = PoiExcelUtil.writeExcelAll("xls");
    response.setContentType("application/excel;charset=utf-8");
    response.setHeader("Content-Disposition", "attachment;filename=" +
excel.getName());
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    response.setDateHeader("Expires", -1);
    ServletOutputStream servletOutputStream = response.getOutputStream();
    FileInputStream fileInputStream = new FileInputStream(excel);
    BufferedInputStream bufferedInputStream = new
BufferedInputStream(fileInputStream);
    int size = 0;
    byte[] b = new byte[4096];
    while ((size = bufferedInputStream.read(b)) != -1) {
        servletOutputStream.write(b, 0, size);
    }
    servletOutputStream.flush();
    servletOutputStream.close();
    bufferedInputStream.close();
}
```