# 尚马教育 JAVA 高级课程

# Excel与Word操作

文档编号：C11

创建日期： 2017-07-07

最后修改日期：2019-10-14

版 本 号：V3.0

电子版文件名：尚马教育-第三阶段-13.excel与word操作.docx

**文档修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日期 | 更新作者 | 更新说明 | 版本号 |
| 2017-07-30 | 张元林 | 初始版本 | V1.0 |
| 2018-08-01 | 王绍成 | Mybatis版本更新 | V2.0 |
| 2019-08-09 | 冯勇涛 | 课件格式以及课程深度加深 | V3.0 |
| 2021-02-26 | 冯勇涛 | 添加maven依赖 | V3.5 |

目录

[1. 常用场景](#_Toc29968_WPSOffice_Level1) [2](#_Toc29968_WPSOffice_Level1)

[2. Excel操作](#_Toc3104_WPSOffice_Level1) [4](#_Toc3104_WPSOffice_Level1)

[2.1. 组件介绍](#_Toc3104_WPSOffice_Level2) [4](#_Toc3104_WPSOffice_Level2)

[2.1.1. JXL](#_Toc3104_WPSOffice_Level3) [4](#_Toc3104_WPSOffice_Level3)

[2.1.2. POI](#_Toc28645_WPSOffice_Level3) [5](#_Toc28645_WPSOffice_Level3)

[2.2. Excel认识](#_Toc28645_WPSOffice_Level2) [6](#_Toc28645_WPSOffice_Level2)

[2.3. 读Excel](#_Toc18137_WPSOffice_Level2) [7](#_Toc18137_WPSOffice_Level2)

[2.3.1. 基本思路](#_Toc18137_WPSOffice_Level3) [7](#_Toc18137_WPSOffice_Level3)

[2.3.2. 代码演示](#_Toc5159_WPSOffice_Level3) [7](#_Toc5159_WPSOffice_Level3)

[2.4. 写Excel](#_Toc5159_WPSOffice_Level2) [9](#_Toc5159_WPSOffice_Level2)

[2.4.1. 代码思路](#_Toc27371_WPSOffice_Level3) [9](#_Toc27371_WPSOffice_Level3)

[2.4.2. Workbook对象](#_Toc19206_WPSOffice_Level3) [9](#_Toc19206_WPSOffice_Level3)

[2.4.3. 代码演示](#_Toc23597_WPSOffice_Level3) [9](#_Toc23597_WPSOffice_Level3)

[3. Word操作](#_Toc28645_WPSOffice_Level1) [11](#_Toc28645_WPSOffice_Level1)

[3.1. 写word](#_Toc27371_WPSOffice_Level2) [11](#_Toc27371_WPSOffice_Level2)

[3.1.1. 基本思路](#_Toc14400_WPSOffice_Level3) [11](#_Toc14400_WPSOffice_Level3)

[3.1.2. 编辑docx模板](#_Toc18849_WPSOffice_Level3) [12](#_Toc18849_WPSOffice_Level3)

[3.1.3. 模板另存为xml](#_Toc26656_WPSOffice_Level3) [12](#_Toc26656_WPSOffice_Level3)

[3.1.4. 代码演示](#_Toc2671_WPSOffice_Level3) [13](#_Toc2671_WPSOffice_Level3)

## 常用场景

|  |
| --- |
| Office软件的读写操作时未来开发软件或者办公系统会经常用到的技术。 |
| **excel读写场景：** |
| 由于公司办公系统或者政府的政务系统等，面向的人群是使用office软件非常熟悉的人群，对于他们来讲，程序员辛辛苦苦开发出来的系统，看似功能强大，在批量录入数据操作习惯性方面，远没有日常使用的office来的熟悉，更希望在Excel中把的数据编辑好，然后导入Excel。  所以，经常会遇见这种情况：为使用者开发好了很好很强大的功能，但是他会提出一个Excel导入数据和导出数据的功能。 |
| **Word使用场景：** |
| 政务系统、教育系统经常用到的技术  公司项目合同导出，用于打印签字。  教育系统问卷导出，用于打印答题。  其他需要打印的场景，word的导出相对于Excel，使用的概率更少。  这里介绍几种Word的导出方式  没有介绍Word的导入方式，实际上不会用word来进行导入，因为数据格式十分不可控。 |

## Excel操作

### 组件介绍

Java对Excel的支持有两个类库：POI和JXL，主流使用POI。

本课程学习POI库。

#### JXL

|  |
| --- |
| * 下载地址：   https://sourceforge.net/projects/jxl/   * 优点：   + 对中文支持非常好，操作简单，方法看名知意。   + 是纯javaAPI，在跨平台上表现的非常完美，代码可以再windows或者Linux上运行而无需重新编写   + 支持Excel 95-2000的所有版本   + 生成Excel 2000标准格式   + 支持字体、数字、日期操作   + 能够修饰单元格属性   + 支持图像和图表,但是这套API对图形和图表的支持很有限，而且仅识别PNG格式 * 缺点：   效率低，图片支持不完善，对格式的支持不如POI强大。  只能读取xls文件，不能读取xlsx文件 |

#### POI

|  |
| --- |
| * 下载地址：   https://poi.apache.org/download.html   * 优点：   + 支持公式，宏，能保持Excel里原有的宏, 但不能用它写新的宏   + 能够修饰单元格属性   + 支持字体、数字、日期操作   + 在一些业务场景中代码相对复杂，但是API丰富，支持多种模式的读写   + 支持比较新版本的excel. * 缺点： * 不支持跨平台（主要就是Java语言） * 读写的时候比较占内存。 * 支持大数量大文件的读写操作   总体来说，对于简单的单表excel导入导出的需求，建议使用JXL。数据量稍微小点，占用内存少，速度快。  对于报表类的，涉及月份数据量，多表数据聚合在一起建议使用POI。 |

### Excel认识

|  |
| --- |
| 学习之前，对Excel的一些知识做一下说明   * 工作簿 Workbook * 工作表 Sheet * 单元格 Cell   一个Excel是由一个工作簿组成  一个工作簿由多个工作表组成  一个工作表是由若干行组成  一个工作表/行也是由若干个单元格组成 |

### 添加poi依赖

Maven依赖

|  |
| --- |
| <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi-ooxml</**artifactId**>  <**version**>3.17</**version**> </**dependency**> <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi-scratchpad</**artifactId**>  <**version**>3.17</**version**> </**dependency**> <**dependency**>  <**groupId**>org.apache.poi</**groupId**>  <**artifactId**>poi-excelant</**artifactId**>  <**version**>3.17</**version**> </**dependency**> |

添加jar包

|  |
| --- |
|  |

### 读Excel

#### 基本思路

* + 获取工作簿
  + 获取工作表
  + 获取总行数
  + 根据第一行获取总列数
  + 根据行数和列数，循环获取每一个单元格
  + 判断单元格数据类型，输出数据

#### 代码演示

|  |
| --- |
| public static void readExcel(){  try {  String excelUrl = "D:\\abc.xlsx";//文件地址  File xlsFile = new File(excelUrl);//创建file对象  //获得工作簿 org.apache.poi.ss.usermodel.Workbook  Workbook workbook = WorkbookFactory.create(xlsFile);  // 获得工作表个数  int sheetCount = workbook.getNumberOfSheets();  // 遍历工作表  for (int i = 0; i < sheetCount; i++) {  //获得工作表org.apache.poi.ss.usermodel.Sheet  Sheet sheet = workbook.getSheetAt(i);  // 获得行数  int rows = sheet.getLastRowNum() + 1;  if(rows <= 1){ continue; }  // 获得列数，先获得一行，在得到该行列数  Row tmp = sheet.getRow(0);  if (tmp == null) { continue; }  //第一行的列数 认为第一行的列数代表整个工作表的列数  int cols = tmp.getPhysicalNumberOfCells();  for (int row = 0; row < rows; row++) {// 读取数据  Row r = sheet.getRow(row);//行  for (int col = 0; col < cols; col++) {  Cell cell = r.getCell(col);//单元格  int type = cell.getCellType();//单元格中的数据类型  String str = "";  switch (type){  case Cell.CELL\_TYPE\_STRING://字符串型 返回值String  str = cell.getStringCellValue();  break;  case Cell.CELL\_TYPE\_NUMERIC://数字型返回值 double  str = String.valueOf(cell.getNumericCellValue());  break;  }  System.out.print(str + "\t");  }  System.out.println();  }  }  } catch (IOException e) {  e.printStackTrace();  } catch (InvalidFormatException e) {  e.printStackTrace();  }  } |

### 写Excel

#### 代码思路

* + 创建输入文件对象FileOutputStream
  + 创建工作簿Workbook
  + 创建工作表Sheet
  + 创建行Row
  + 创建单元格Cell
  + 写入数据

#### Workbook对象

该对象有三个实现类：

* + HSSFWorkbook 是操作Excel2003以前（包括2003）的版本，扩展名是.xls。该对象生成excel有所限制，导出的行数至多为65535行，超出65536条后系统就会报错。
  + XSSFWorkbook 是 操作Excel2007后的版本，扩展名是.xlsx。
  + 突破HSSFWorkbook的65535行局限，最多可以导出104万行，仍然存在问题---OOM内存溢出，原因是创建的workbook sheet row cell对象是在内存的，并没有持久化。
  + SXSSFWorkbook是 操作Excel2007后的版本，扩展名是.xlsx。
  + 对于大型excel文件的创建，要确保不会内存溢出，只有使用SXSSFWorkbook，该对象用硬盘空间换内存，避免内存溢出。

#### 代码演示

|  |
| --- |
| public static void writeExcel(){  try {  String excelUrl = "D:\\javasm.xlsx";//导出地址  File excelFile = new File(excelUrl);//创建File对象  FileOutputStream xlsStream = new FileOutputStream(excelFile);  // 创建工作薄 org.apache.poi.hssf.usermodel.HSSFWorkbook  HSSFWorkbook workbook = new HSSFWorkbook();  // 创建工作表 org.apache.poi.hssf.usermodel.HSSFSheet  HSSFSheet sheet = workbook.createSheet("工作表名称");  //循环向工作表(Sheet)中添加单元格(Cell)  String temple = "第%s行%s列";  //假设添加 10行 8列的数据  for (int row = 0; row < 10; row++) {  //创建一行 org.apache.poi.hssf.usermodel.HSSFRow  HSSFRow rows = sheet.createRow(row);  for (int col = 0; col < 8; col++) {  //org.apache.poi.ss.usermodel.Cell  Cell cell = rows.createCell(col);//单元格  // 向工作表中添加数据  cell.setCellValue(String.format(temple,row,col));  }  }  workbook.write(xlsStream);  } catch (IOException e) {  e.printStackTrace();  }  } |

## Word操作

### 写word

关于word的导出，方法非常多，本课程只学习最常用方法：基于freeMarker模板引擎进行word导出。

|  |
| --- |
| Freemarker不是专门为导出word而设计的技术。  FreeMarker是一款模板引擎，多用于生成HTML页面，源代码等，是一种基于模板改变数据生成文本的工具。  FreeMarker是开源的技术，有自己独特的语法结构  在线文档：http://freemarker.foofun.cn/ref.html  文档中有较为详细的教程语法介绍 |

#### 添加freemarker依赖

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/org.freemarker/freemarker -->  <dependency>  <groupId>org.freemarker</groupId>  <artifactId>freemarker</artifactId>  <version>2.3.23</version>  </dependency> |

#### 基本思路

* 先编辑模板，这是最重要的，编辑docx，另存为xml
* 模板中可以有list，判断，map等
* Java中拼装数据，存入map
* Map中的key，是xml模板中需要读取的变量
* 模板路径和名称设置正确
* 设置好编码
* 写入

#### 编辑docx模板

|  |
| --- |
|  |

#### 模板另存为xml

|  |
| --- |
|  |

#### 代码演示

|  |
| --- |
| public void writeWord(){  try {  Map<String,Object> map = new HashMap<>();  map.put("userName","fyt");  map.put("userCode","41012345646645821222");  map.put("housePrice","$2000");  map.put(“houseDate”,”2019-10-14”);  Writer out = new OutputStreamWriter(new FileOutputStream("F:\\test10.docx"), "UTF-8");  //freemarker.template.Configuration  Configuration configuration = new Configuration();  configuration.setDefaultEncoding("UTF-8");//设置编码  //类加载器，设置xml模板文件的路径  configuration.setClassForTemplateLoading(this.getClass(), "/com/javasm/word/xml");  //获取模板对象 传入模板名称  Template t = configuration.getTemplate("test.xml");  t.process(map, out);//模板中传入数据  out.close();  } catch (IOException | TemplateException e) {  e.printStackTrace();  }  } |