java POI常用API

# poi常用API

|  |
| --- |
| //获取工作薄对象:  HSSFWorkbook wb = new HSSFWorkbook();  //创建sheet  HSSFSheet sheet = wb.createSheet([String name]);  //创建单元格并设置值  row.createCell(16).setCellValue(empName);  row.createCell(15).setCellType(HSSFCell.CELL\_TYPE\_BLANK); //设置空值  //设置单元格公式  Cell tailCell1 = countRow.createCell(6);  tailCell1.setCellType(HSSFCell.CELL\_TYPE\_FORMULA);  tailCell1.setCellFormula("SUM(G3:G" + rowNum + ")");  //合并单元格  Region region1 = new Region(0, (short) 0, 0, (short) 6); //参数1：行号 参数2：起始列号 参数3：行号 参数4：终止列号  CellRangeAddress region1 = new CellRangeAddress(rowNumber, rowNumber, (short) 0, (short) 11);  //使用方法  Cell cell=row.createCell(1);  cell.setCellValue("我们被合并单元格啦！");  sheet.addMergedRegion(new CellRangeAddress(  1, // 起始行  2, // 结束行  1, // 其实列  2 // 结束列  ));  CellRangeAddress cra1 = new CellRangeAddress(rowNum, rowNum + rowCount, 0, 0);  sheet.addMergedRegion(cra1); |
| //设置单元格格式  HSSFCellStyle cellStyle = wb.createCellStyle();  CellStyle cellStyle=wb.createCellStyle(); // 创建单元格样式  //设置背景色  cellStyle.setFillForegroundColor((short) 13);// 设置背景色  cellStyle.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND);  //设置边框  cellStyle.cellStyleBottom(HSSFCellStyle.BORDER\_THIN); //下边框  cellStyle.cellStyleLeft(HSSFCellStyle.BORDER\_THIN);//左边框  cellStyle.cellStyleTop(HSSFCellStyle.BORDER\_THIN);//上边框  cellStyle.cellStyleRight(HSSFCellStyle.BORDER\_THIN);//右边框  //设置单元格中数据对其方式  cellStyle.setAlignment(HSSFCellStyle.ALIGN\_CENTER); // 居中  //设置字体  HSSFFont font = wb.createFont();  font.setFontName("黑体");  font.setFontHeightInPoints((short) 16);//设置字体大小  HSSFFont font2 = wb.createFont();  font2.setFontName("仿宋\_GB2312");  font2.setBoldweight(HSSFFont.BOLDWEIGHT\_BOLD);//粗体显示  font2.setFontHeightInPoints((short) 12);  cellStyle.setFont(font);//选择需要用到的字体格式  //设置列宽  sheet.setColumnWidth(24, 12\*256); //第一个参数代表列id(从0开始),第2个参数代表宽度值  //设置自动换行  cellStyle.setWrapText(true);  //单元格设置样式  cell.setCellStyle(cellStyle); |

# excel导出例子

|  |
| --- |
| @RequestMapping(value = "/export", method = RequestMethod.***GET***)  **public** **void** toExcle(HttpServletRequest request, HttpServletResponse response, EmployeeIncomePageDto pageDto)  **throws** UnsupportedEncodingException {  HSSFWorkbook wb = employeeIncomeService.toExcle(pageDto);  String excelName = "服务收入详情表.xls";  **try** {  response.setContentType("application/vnd.ms-excel;charset=utf-8");  response.setHeader("Content-disposition",  "attachment;filename=" + **new** String(excelName.getBytes("GB2312"), "ISO8859-1"));  ServletOutputStream outputStream = response.getOutputStream();  wb.write(outputStream);  outputStream.flush();  } **catch** (IOException e) {  ***logger***.error(e.getMessage());  }  } |
| public class ExcelUtil {  public static HSSFWorkbook buildExcelWorkBook(String sheetName, String[] header, List<Object[]> datas,  int[] disMergeCols) {  HSSFWorkbook workBook = new HSSFWorkbook();  CellStyle headerCellStyle = createCellStyle(workBook, true);  CellStyle bodyCellStyle = createCellStyle(workBook, false);  buildExcelHeader(workBook, sheetName, header, headerCellStyle);  buildExcelBody(workBook, datas, disMergeCols, bodyCellStyle);  return workBook;  }  private static void buildExcelHeader(HSSFWorkbook workBook, String sheetName, String[] header,  CellStyle headerCellStyle) {  //创建sheet  HSSFSheet sheet = workBook.createSheet(sheetName);  HSSFRow headRow = sheet.createRow(0);  for (int i = 0; i < header.length; i++) {  //设置整体列格式  sheet.setColumnWidth(i, 16 \* 256);  HSSFCell cell = headRow.createCell(i);  cell.setCellValue(header[i].toString());  cell.setCellStyle(headerCellStyle);  }  }  private static void buildExcelBody(HSSFWorkbook workBook, List<Object[]> datas, int[] disMergeCols,  CellStyle bodyCellStyle) {  HSSFSheet sheet = workBook.getSheetAt(0);  for (int i = 0; i < datas.size(); i++) {  HSSFRow row = sheet.createRow(i + 1);  Object[] objs = datas.get(i);  for (int j = 0; j < objs.length; j++) {  if (!ArrayUtils.contains(disMergeCols, j) && i != 0  && objs[0].toString().equals(datas.get(i - 1)[0].toString())) {  HSSFCell cell = row.createCell(j);  cell.setCellStyle(bodyCellStyle);  CellRangeAddress range = new CellRangeAddress(i, i + 1, j, j);  sheet.addMergedRegion(range);  } else {  HSSFCell cell = row.createCell(j);  if (objs[j] != null) {  cell.setCellValue(objs[j].toString());  }  cell.setCellStyle(bodyCellStyle);  }  }  }  }  private static HSSFCellStyle createCellStyle(HSSFWorkbook workBook, boolean isHeader) {  HSSFCellStyle cellStyle = workBook.createCellStyle();  cellStyle.setBorderBottom(HSSFCellStyle.BORDER\_THIN); //下边框  cellStyle.setBorderLeft(HSSFCellStyle.BORDER\_THIN);//左边框  cellStyle.setBorderTop(HSSFCellStyle.BORDER\_THIN);//上边框  cellStyle.setBorderRight(HSSFCellStyle.BORDER\_THIN);//右边框  //自动换行  cellStyle.setWrapText(true);  //设置字体  HSSFFont font = workBook.createFont();  font.setBoldweight(HSSFFont.BOLDWEIGHT\_BOLD);//粗体显示  //表头的样式  if (isHeader) {  cellStyle.setFillForegroundColor((short) 13);// 设置背景色  cellStyle.setFillPattern(HSSFCellStyle.SOLID\_FOREGROUND);  cellStyle.setAlignment(HSSFCellStyle.ALIGN\_CENTER);  //表头字体设置  cellStyle.setFont(font);  }  return cellStyle;  }  } |
| **public** **class** ExcelUtil {  **public** **static** HSSFWorkbook buildExcelWorkBook(String sheetName, String[] header, List<Object[]> datas,  **int**[] disMergeCols) {  HSSFWorkbook workBook = **new** HSSFWorkbook();  CellStyle headerCellStyle = *createCellStyle*(workBook, **true**);  CellStyle bodyCellStyle = *createCellStyle*(workBook, **false**);  *buildExcelHeader*(workBook, sheetName, header, headerCellStyle);  **if** (datas != **null**) {  *buildExcelBody*(workBook, datas, disMergeCols, bodyCellStyle);  }  **return** workBook;  }  **private** **static** **void** buildExcelHeader(HSSFWorkbook workBook, String sheetName, String[] header,  CellStyle headerCellStyle) {  // 创建sheet  HSSFSheet sheet = workBook.createSheet(sheetName);  HSSFRow headRow = sheet.createRow(0);  **for** (**int** i = 0; i < header.length; i++) {  // 设置整体列格式  sheet.setColumnWidth(i, 16 \* 256);  HSSFCell cell = headRow.createCell(i);  cell.setCellValue(header[i].toString());  cell.setCellStyle(headerCellStyle);  }  }  **private** **static** **void** buildExcelBody(HSSFWorkbook workBook, List<Object[]> datas, **int**[] disMergeCols,  CellStyle bodyCellStyle) {  HSSFSheet sheet = workBook.getSheetAt(0);  **for** (**int** i = 0; i < datas.size(); i++) {  HSSFRow row = sheet.createRow(i + 1);  Object[] objs = datas.get(i);  //是否是重复行  **boolean** isRepeatRow = i != 0 && objs[0].toString().equals(datas.get(i - 1)[0].toString());  **for** (**int** j = 0; j < objs.length-1; j++) {  **if** (disMergeCols != **null** && !ArrayUtils.*contains*(disMergeCols, j) && isRepeatRow) {  HSSFCell cell = row.createCell(j);  cell.setCellStyle(bodyCellStyle);  CellRangeAddress range = **new** CellRangeAddress(i, i + 1, j, j);  sheet.addMergedRegion(range);  } **else** {  HSSFCell cell = row.createCell(j);  //objs[0]是比较数据  **if** (objs[j+1] != **null**) {  cell.setCellValue(objs[j+1].toString());  }  cell.setCellStyle(bodyCellStyle);  }  }  }  }  **private** **static** HSSFCellStyle createCellStyle(HSSFWorkbook workBook, **boolean** isHeader) {  HSSFCellStyle cellStyle = workBook.createCellStyle();  cellStyle.setBorderBottom(HSSFCellStyle.***BORDER\_THIN***); // 下边框  cellStyle.setBorderLeft(HSSFCellStyle.***BORDER\_THIN***);// 左边框  cellStyle.setBorderTop(HSSFCellStyle.***BORDER\_THIN***);// 上边框  cellStyle.setBorderRight(HSSFCellStyle.***BORDER\_THIN***);// 右边框  // 自动换行  cellStyle.setWrapText(**true**);  // 设置字体  HSSFFont font = workBook.createFont();  font.setBoldweight(HSSFFont.***BOLDWEIGHT\_BOLD***);// 粗体显示  // 表头的样式  **if** (isHeader) {  cellStyle.setFillForegroundColor((**short**) 13);// 设置背景色  cellStyle.setFillPattern(HSSFCellStyle.***SOLID\_FOREGROUND***);  cellStyle.setAlignment(HSSFCellStyle.***ALIGN\_CENTER***);  // 表头字体设置  cellStyle.setFont(font);  }  **return** cellStyle;  }  } |