Java 操作 Excel

参考：慕课网 - 解密 JAVA 实现 Excel 导入导出
制作日期：2018-08-01
制作人：小桅[yw_forgit@163.com]

1、读取 Excel 有三种常用技术

读取 Excel 有三种常用技术：POI、JXL、FASTEXCEL

1）POI

Apache POI 是 Apache 软件基金会的开放源码函式库,POI 提供 API 给 Java 程序对 Microsoft Office 格式档案读和写的功能。

HSSF 是 Horrible SpreadSheet Format 的缩写，也即“讨厌的电子表格格式”。通过 HSSF，你可以用纯 Java 代码来读取、写入、修改 Excel 文件。

doc 是 office2003 及以前版本 word 的文件，docx 是 office2007 及以后版本 word 的文件，其他同理。

HSSF - 读写 Microsoft Excel 格式档案的功能，即 xls 后缀。
XSSF - 读写 Microsoft Excel OOXML 格式档案的功能，即 xlsx 后缀。

HWPF - 读写 Microsoft Word 格式档案的功能，即 doc 后缀。
HSLF - 读写 Microsoft PowerPoint 格式档案的功能，即 ppt 后缀。
HDGF - 读写 Microsoft Visio 格式档案的功能，即 vsd 后缀。
（应该后缀是带 x 的都是使用 X...来读写的吧）

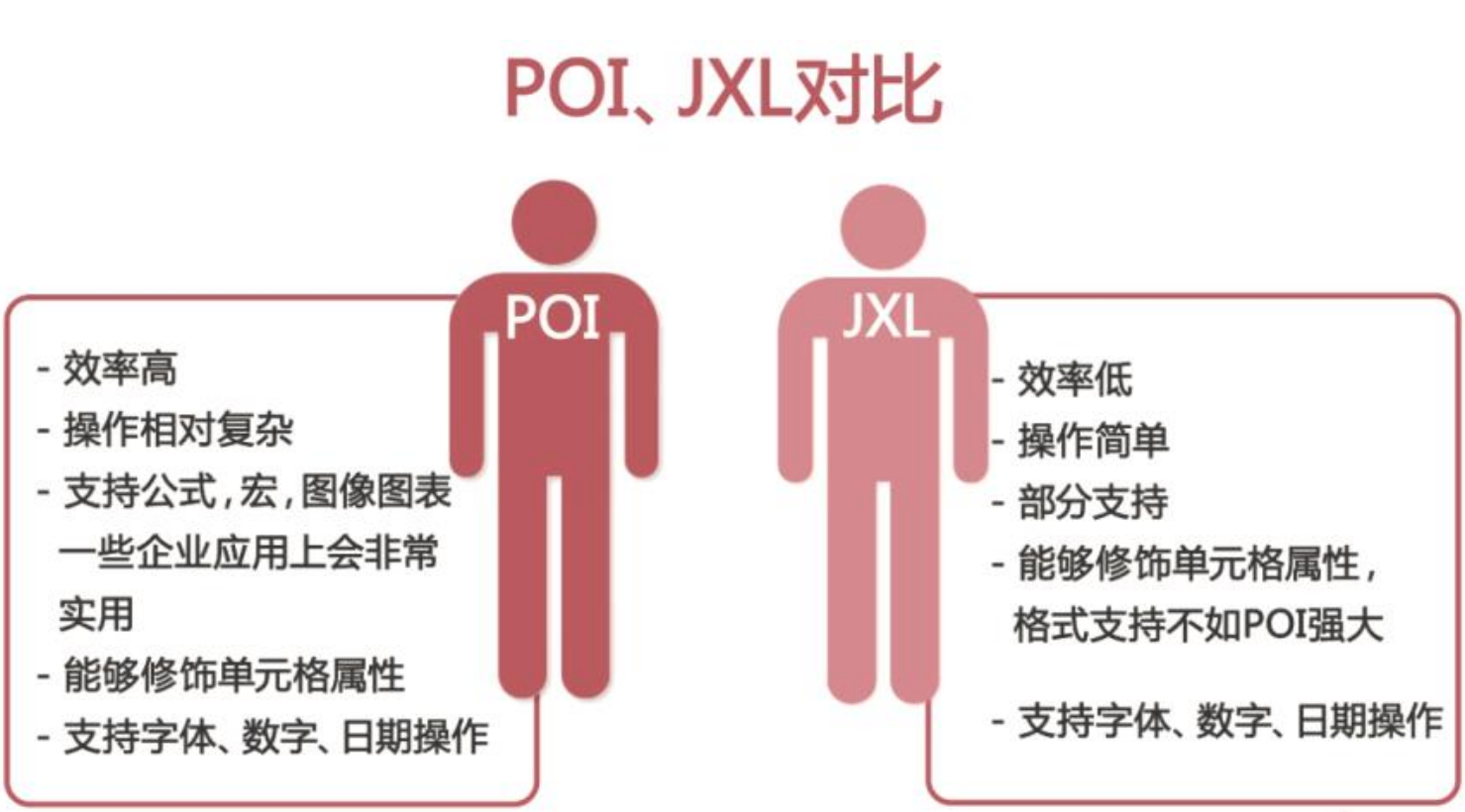
iText

在这里提一下。
通过 iText 不仅可以生成 PDF 或 rtf 的文档，而且可以将 XML、Html 文件转化为 PDF 文件。
下载 iText.jar 文件后，只需要在系统的 CLASSPATH 中加入 iText.jar 的路径，在程序中就可以使用 iText 类库了。

2）JXL

Java Excel 是一开放源码项目，可以读取 Excel 文件的内容、创建新的 Excel 文件、更新已经存在的 Excel 文件。包括常见格式的设置：字体,颜色、背景、合并单元格等。

POI 与 JXL 对比：



3）FASTEXCEL

比较少用。
FastExcel 是一个采用纯 java 开发的 excel 文件洟写組件，支持 Excel 97-2003 版本的文件格式，即 xls 后缀。
FastExce 只能读取单元格的字符信息，而其它属性如颜色、字体等就不支持了，因此 FastExce 只需很小的内存。

2、对 Excel 的一些认知

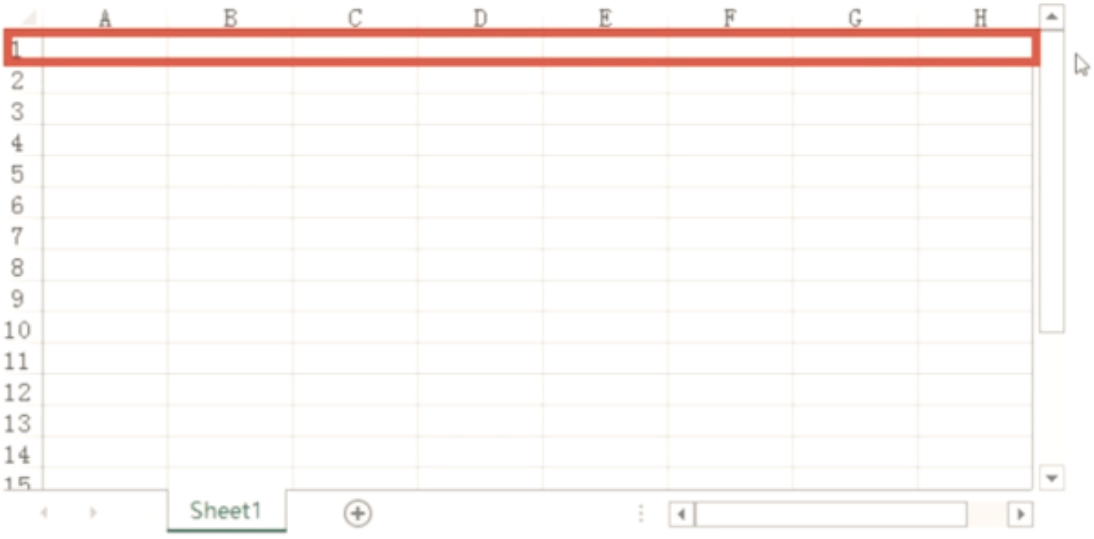
工作簿

工作表

工作表 Sheet

行

行记录 Row



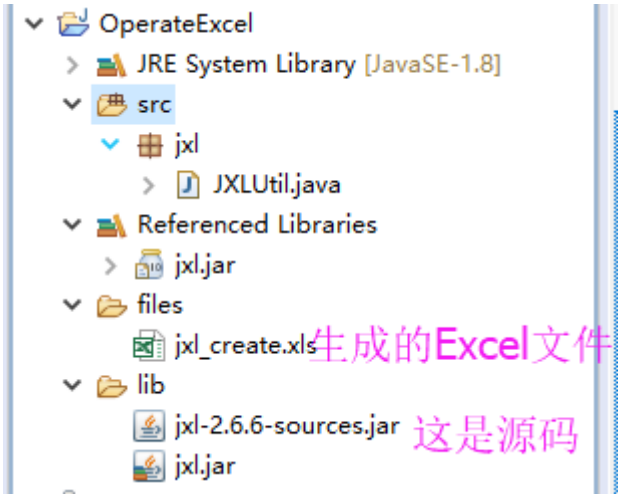
单元格

单元格Cell

3、JXL 操作 Excel

代码

目录结构，还要引入 JUINT4，这里没有截图到



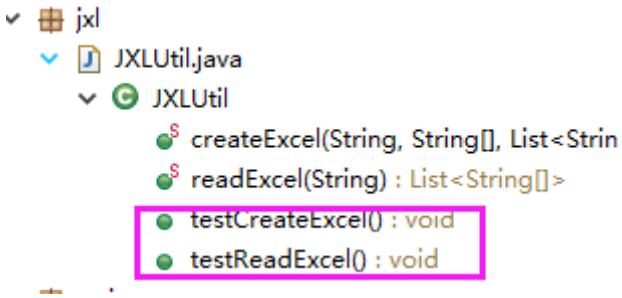
JXLUtil.java

```
package jxl;

import java.io.File;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;


import org.junit.Test;


import jxl.read.biff.BiffException;

import jxl.write.Label;

import jxl.write.WritableSheet;

import jxl.write.WritableWorkbook;

import jxl.write.WriteException;

import jxl.write.biff.RowsExceededException;


public class JXLUtil {


	@Test

	public void testCreateExcel() {

		// System.getProperty("user.dir")获取项目路径。E:\Code\EclipseWorkspace\MyTools\OperateExcel

		String filePath = System.getProperty("user.dir") + "/files/jxl_create.xls"; // xls是2003版以及以前的版本

		// 尽管输出结果是E:\Code\EclipseWorkspace\MyTools\OperateExcel/files/jxl_create.xls，带\和/,在Windows下也可以访问到文件的

		// System.out.println(filePath);

		String[] dataHead = new String[]{"编号", "名称", "性别"};

		// 将三个List自动组装成list

		List<String[]> dataList = Arrays.asList(

				new String[]{"1", "张三", "男"},

				new String[]{"2", "李四", "女"},
```

```
                new String[]{"3", "小五", "女"},
                new String[]{"4", "老六", "男"});
        createExcel(filePath, dataHead, dataList);
        System.out.println("jxl创建完成");
    }

    @Test
    public void testReadExcel() {
        // System.getProperty("user.dir")获取项目路径。E:\Code\EclipseWorkspace\MyTools\OperateExcel
        String filePath = System.getProperty("user.dir") + "/files/jxl_create.xls"; // xls是2003版以及以前的版本
        List<String[]> dataList = readExcel(filePath);
        for (int i = 0; i < dataList.size(); i++) {
            String[] row = dataList.get(i);
            for (int j = 0; j < row.length; j++) {
                System.out.print(row[j] + " ");
            }
            // 一行之后换行
            System.out.println();
        }
    }

    /**
     * 根据行头、数据创建Excel文件
     * @param filePath 文件存放的路径
     * @param dataHead 行头，就是第一行的的标签
     * @param dataList 数据。list大小表示行的大小，每个数组的大小表示列的大小
     */
    public static void createExcel(String filePath, String[] dataHead, List<String[]> dataList) {
        try {
            // >> 创建xls格式文件
            File file = new File(filePath);
            file.createNewFile();
            // >> 创建工作簿
            WritableWorkbook workbook = Workbook.createWorkbook(file);
            // >> 创建工作表
            WritableSheet sheet = workbook.createSheet("sheet0", 0); // sheet名称、第几个sheet
            Label label = null;
            // >> 添加数据头
            for (int i = 0; i < dataHead.length; i++) {
                // 第i列，第0行（即第一行行头），dataHead[i]就是具体的内容
                label = new Label(i, 0, dataHead[i]);
                // Cell就是单元格，添加一个单元格
                sheet.addCell(label);
            }
            // >> 追加数据
            // 第几行，List大小
            for (int i = 0; i < dataList.size(); i++) {
                // 第几列，行头大小
                for (int j = 0; j < dataHead.length; j++) {
                    // 第j列，第i + 1行（第一行给行头了），dataList.get(i)[j]先j就是先获取行（list大小表示行的大小）再获取列（每个数组的大小表示列的大小）
                    label = new Label(j, i + 1, dataList.get(i)[j]);
                    // 添加到单元格
```

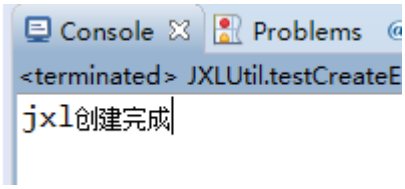
```
                        sheet.addCell(label);
                    }
                }
                // >> 写出数据并关闭
                workbook.write();
                workbook.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (RowsExceededException e) {
                e.printStackTrace();
            } catch (WriteException e) {
                e.printStackTrace();
            }
        }

        /**
         * 从一个Excel文件中，读取内容
         * @param filePath
         * @return 类型是List<String[]>，list大小表示行的大小，每个数组的大小表示列的大小
         */
        public static List<String[]> readExcel(String filePath) {
            List<String[]> dataList = new ArrayList<String[]>();
            try {
                // 》》 获取工作簿，注意不是创建，否则没有数据的哦，并且要确保这文件有数据哦
                Workbook workbook = Workbook.getWorkbook(new File(filePath));
                // 》》 获取工作簿
                Sheet sheet = workbook.getSheet(0);
                // 》》 获取数据
                // 行要在最外层，列在里头
                for (int i = 0; i < sheet.getRows(); i++) {
                    // 一行数据的内容
                    String[] rowContens = new String[sheet.getColumns()];
                    for (int j = 0; j < sheet.getColumns(); j++) {
                        // j列，i行
                        Cell cell = sheet.getCell(j, i);
                        // 单元格内容
                        String content = cell.getContents();
                        rowContens[j] = content;
                    }
                    dataList.add(rowContens);
                }
                // 》》 关闭
                workbook.close();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (BiffException e) {
                e.printStackTrace();
            }
            return dataList;
        }

}
```

两个测试方法

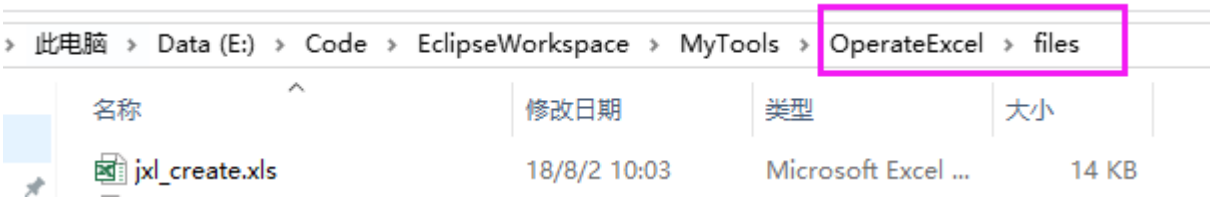


创建 Excel 文件

执行 testCreateExcel()测试方法，创建 Excel 文件



打开 jxl_create.xls 文件看下

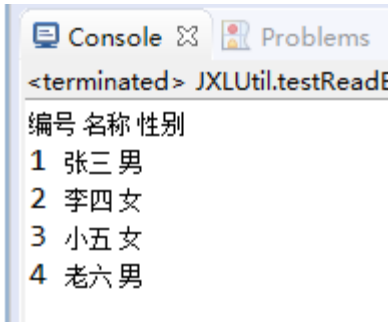


内容如下

	A	B	C
1	编号	名称	性别
2	1	张三	男
3	2	李四	女
4	3	小五	女
5	4	老六	男

读取 Excel 文件

执行 testReadExcel()测试方法，读取 Excel 文件，输出结果如下

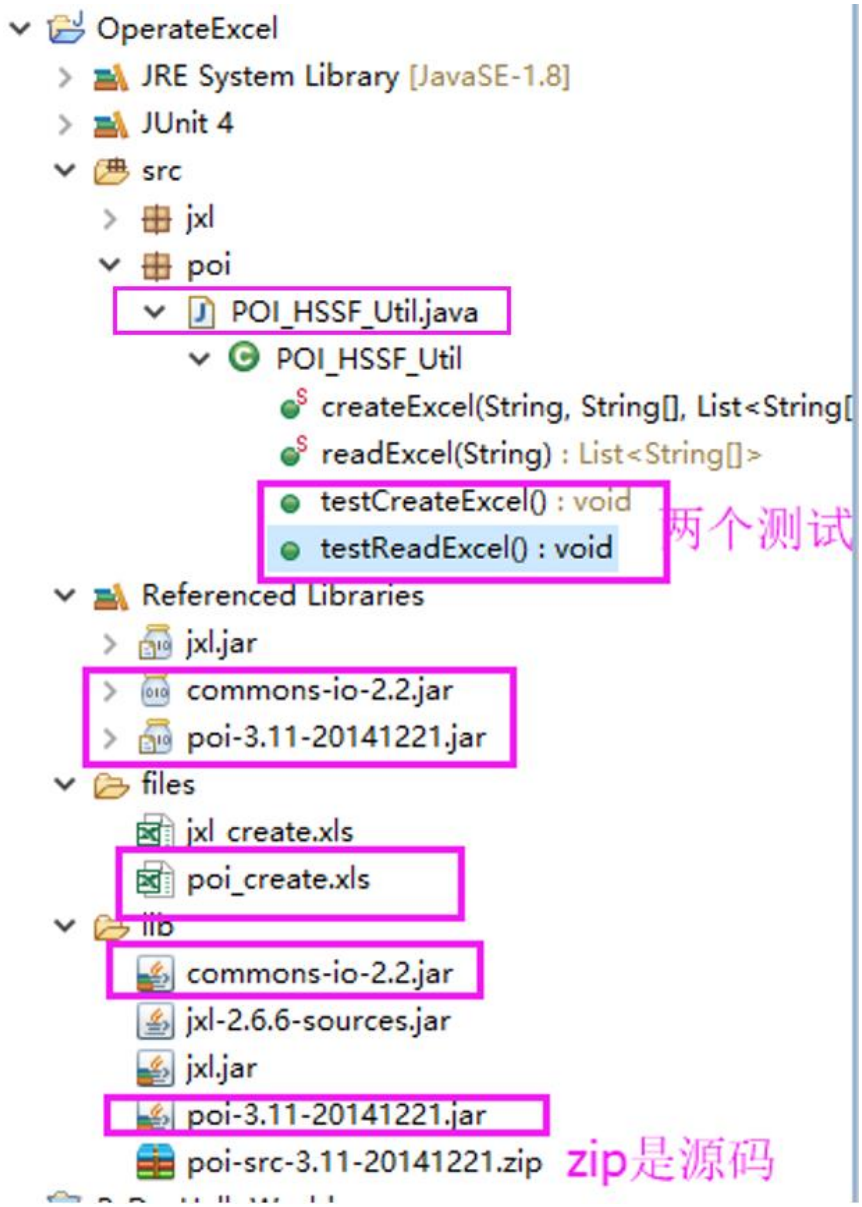


4、POI 操作 Excel

1）HSSF 操作 xls（2003 版本及以前的）

环境是接着上一节的，紫色部分表示是当前的

代码



POI_HSSF_Util.java

```
package poi;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import org.apache.commons.io.FileUtils;
```

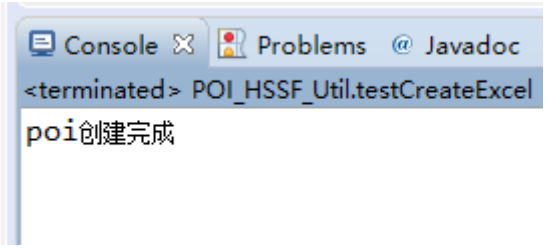
```java
import org.apache.poi.hssf.usermodel.HSSFCell;
import org.apache.poi.hssf.usermodel.HSSFRow;
import org.apache.poi.hssf.usermodel.HSSFSheet;
import org.apache.poi.hssf.usermodel.HSSFWorkbook;
import org.junit.Test;

public class POI_HSSF_Util {

    @Test
    public void testCreateExcel() {
        // System.getProperty("user.dir")获取项目路径。E:\Code\EclipseWorkspace\MyTools\OperateExcel
        String filePath = System.getProperty("user.dir") + "/files/poi_create.xls"; // xls是2003版以及以前的版本
        String[] dataHead = new String[]{"编号", "名称", "性别"};
        // 将三个List自动组装成list
        List<String[]> dataList = Arrays.asList(
                new String[]{"1", "张三", "男"},
                new String[]{"2", "李四", "女"},
                new String[]{"3", "小五", "女"},
                new String[]{"4", "老六", "男"});
        createExcel(filePath, dataHead, dataList);
        System.out.println("poi创建完成");
    }

    @Test
    public void testReadExcel() {
        // System.getProperty("user.dir")获取项目路径。E:\Code\EclipseWorkspace\MyTools\OperateExcel
        String filePath = System.getProperty("user.dir") + "/files/poi_create.xls"; // xls是2003版以及以前的版本
        List<String[]> dataList = readExcel(filePath);
        for (int i = 0; i < dataList.size(); i++) {
            String[] row = dataList.get(i);
            for (int j = 0; j < row.length; j++) {
                System.out.print(row[j] + " ");
            }
            // 一行之后换行
            System.out.println();
        }
    }

    /**
     * 根据行头、数据创建Excel文件
     * @param filePath 文件存放的路径
     * @param dataHead 行头，就是第一行的的标签
     * @param dataList 数据。list大小表示行的大小，每个数组的大小表示列的大小
     */
    public static void createExcel(String filePath, String[] dataHead, List<String[]> dataList) {
        try {
            // >> 创建工作簿
            HSSFWorkbook workbook = new HSSFWorkbook();
            // >> 创建工作表
            HSSFSheet sheet = workbook.createSheet("sheet0");
            // >> 根据sheet，创建表头，第一行
            HSSFRow row0 = sheet.createRow(0);
```

```
            for (int i = 0; i < dataHead.length; i++) {
                // 根据当前行，创建第i个单元格
                HSSFCell cell = row0.createCell(i);
                // 设置当前单元格的值
                cell.setCellValue(dataHead[i]);
            }
            // >> 追加数据
            for (int i = 0; i < dataList.size(); i++) {
                // 创建第i + 1行
                HSSFRow row = sheet.createRow(i + 1);
                for (int j = 0; j < dataHead.length; j++) {
                    // 创建第i + 1行的第j列（定位到当前单元格）
                    HSSFCell cell = row.createCell(j);
                    // 把第list大小就是第几行，每个数组的大小就是第几列，就是第i+1行第j列的单元格的值
                    cell.setCellValue(dataList.get(i)[j]);
                }
            }
            // >> 输出，并关闭
            File file = new File(filePath);
            file.createNewFile();
            FileOutputStream outputStream = FileUtils.openOutputStream(file);
            workbook.write(outputStream);
            outputStream.close();
            workbook.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * 从Excel文件中，读取内容
     * @param filePath
     * @return 类型是List<String[]>，list大小表示行的大小，每个数组的大小表示列的大小
     */
    public static List<String[]> readExcel(String filePath) {
        List<String[]> dataList = new ArrayList<String[]>();
        try {
            // >> 从输入流创建工作簿
            HSSFWorkbook workbook = new HSSFWorkbook(FileUtils.openInputStream(new File(filePath)));
            // >> 获取第0个工作表
            HSSFSheet sheet = workbook.getSheetAt(0);
            // >> 获取数据
            // 获取最后一行行号
            int lastRowNum = sheet.getLastRowNum();
            // lastRowNum + 1，因为lastRowNum表示当前行是最后一行，但是下标从0开始的
            for (int i = 0; i < lastRowNum + 1; i++) {
                // 获取第i行
                HSSFRow row = sheet.getRow(i);
                // 获取最后一个单元格的列号
                short lastCellNum = row.getLastCellNum();
                // 存储一行的数据
                String[] rowValue = new String[lastCellNum];
```

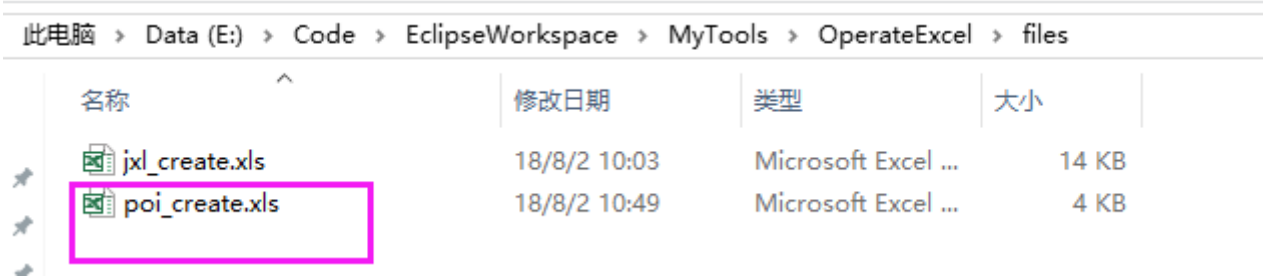
```
                for (int j = 0; j < lastCellNum; j++) {
                    // 获取第i行的第j列的单元格
                    HSSFCell cell = row.getCell(j);
                    // 由于前面放进去的值都是String，故这里以String的形式获取
                    String cellValue = cell.getStringCellValue();
                    rowValue[j] = cellValue;
                }
                dataList.add(rowValue);
            }
            // >> 关闭
            workbook.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return dataList;
    }
}
```

创建 Excel 文件

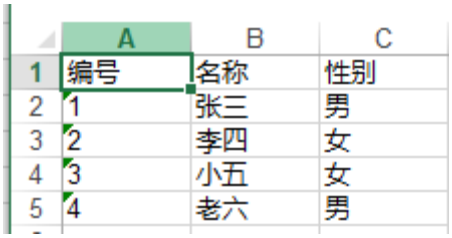
执行 testCreateExcel()测试方法，创建 Excel 文件



打开 poi_create.xls 看下

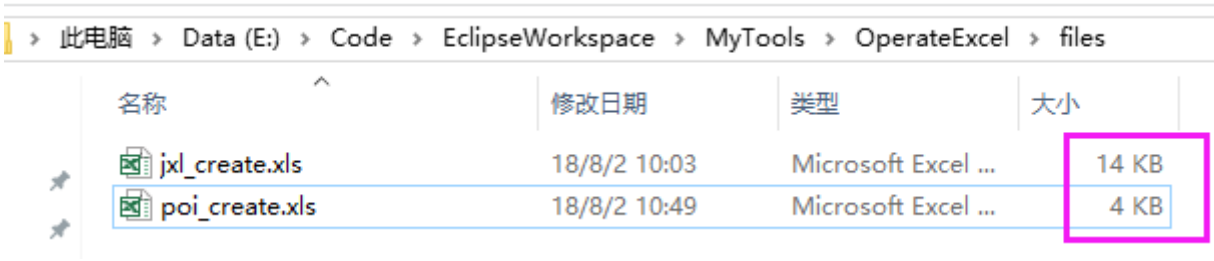


内容



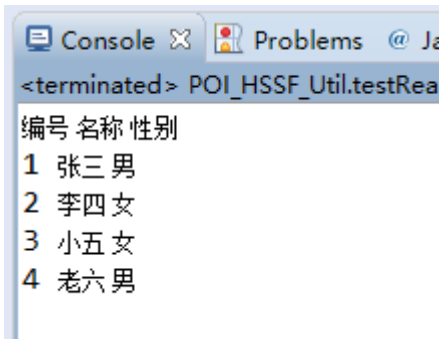
	A	B	C
1	编号	名称	性别
2	1	张三	男
3	2	李四	女
4	3	小五	女
5	4	老六	男

对比一下，发现同样的数据，POI 创建的 Excel 文件大小比较小。



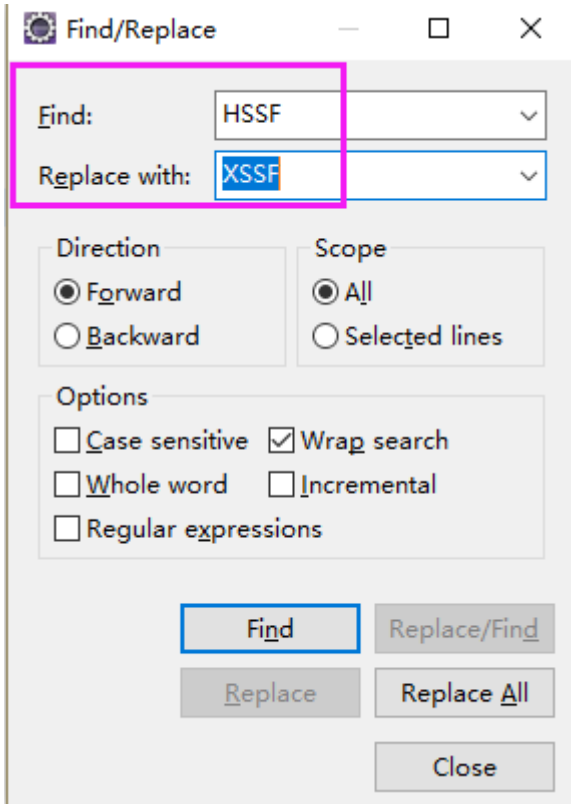
读取 Excel 文件

执行 testReadExcel()测试方法，读取 Excel 文件，输出结果如下



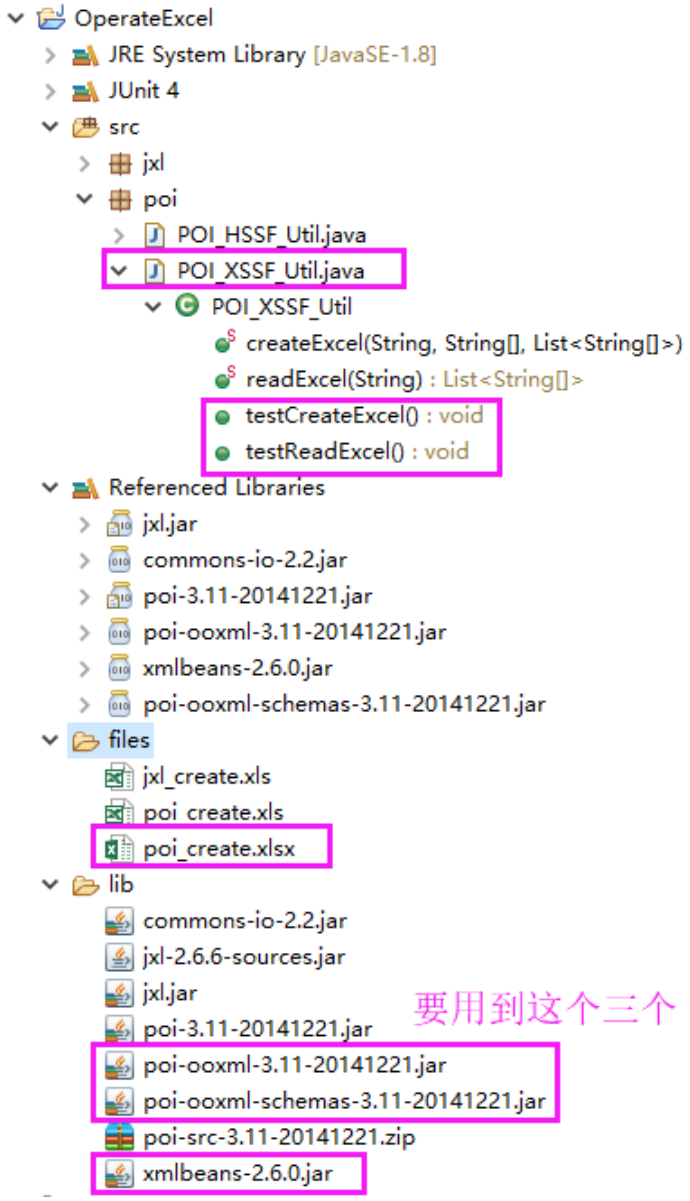
2）XSSF 操作 xlsx（2007 版本及以后的）

很简单，复制上节的 POI_HSSF_Util.java，然后全部把 HSSF 替换为 XSSF，之后导入相关 jar 包，再引入即可。



代码

目录结构



POI_XSSF_Util.java

```
package poi;


import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import org.apache.commons.io.FileUtils;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
import org.junit.Test;

```

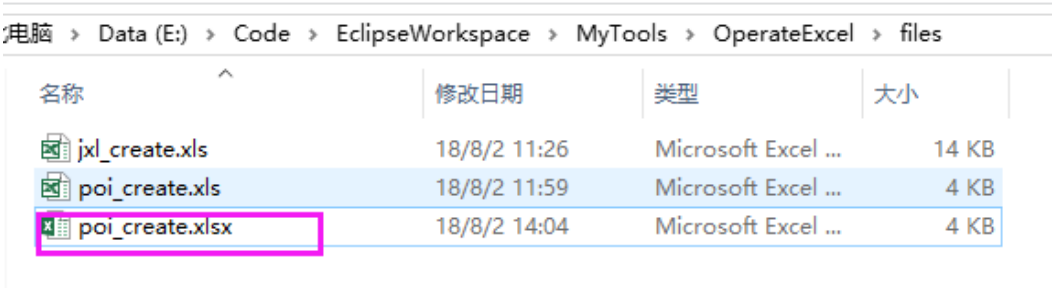
```java
public class POI_XSSF_Util {

    @Test
    public void testCreateExcel() {
        // System.getProperty("user.dir")获取项目路径。E:\Code\EclipseWorkspace\MyTools\OperateExcel
        String filePath = System.getProperty("user.dir") + "/files/poi_create.xlsx"; // xlsx是2007版以及以后的版本
        String[] dataHead = new String[]{"编号", "名称", "性别"};
        // 将三个List自动组装成list
        List<String[]> dataList = Arrays.asList(
                new String[]{"1", "张三", "男"},
                new String[]{"2", "李四", "女"},
                new String[]{"3", "小五", "女"},
                new String[]{"4", "老六", "男"});
        createExcel(filePath, dataHead, dataList);
        System.out.println("poi_xssf_创建完成");
    }

    @Test
    public void testReadExcel() {
        // System.getProperty("user.dir")获取项目路径。E:\Code\EclipseWorkspace\MyTools\OperateExcel
        String filePath = System.getProperty("user.dir") + "/files/poi_create.xlsx"; // xlsx是2007版以及以后的版本
        List<String[]> dataList = readExcel(filePath);
        for (int i = 0; i < dataList.size(); i++) {
            String[] row = dataList.get(i);
            for (int j = 0; j < row.length; j++) {
                System.out.print(row[j] + " ");
            }
            // 一行之后换行
            System.out.println();
        }
    }

    /**
     * 根据行头、数据创建Excel文件
     * @param filePath 文件存放的路径
     * @param dataHead 行头，就是第一行的的标签
     * @param dataList 数据。list大小表示行的大小，每个数组的大小表示列的大小
     */
    public static void createExcel(String filePath, String[] dataHead, List<String[]> dataList) {
        try {
            // >> 创建工作簿
            XSSFWorkbook workbook = new XSSFWorkbook();
            // >> 创建工作表
            XSSFSheet sheet = workbook.createSheet("sheet0");
            // >> 根据sheet，创建表头，第一行
            XSSFRow row0 = sheet.createRow(0);
            for (int i = 0; i < dataHead.length; i++) {
                // 根据当前行，创建第i个单元格
                XSSFCell cell = row0.createCell(i);
                // 设置当前单元格的值
                cell.setCellValue(dataHead[i]);
            }
```

```
            // >> 追加数据
            for (int i = 0; i < dataList.size(); i++) {
                // 创建第i + 1行
                XSSFRow row = sheet.createRow(i + 1);
                for (int j = 0; j < dataHead.length; j++) {
                    // 创建第i + 1行的第j列（定位到当前单元格）
                    XSSFCell cell = row.createCell(j);
                    // 把第list大小就是第几行，每个数组的大小就是第几列，就是第i+1行第j列的单元格的值
                    cell.setCellValue(dataList.get(i)[j]);
                }
            }
            // >> 输出，并关闭
            File file = new File(filePath);
            file.createNewFile();
            FileOutputStream outputStream = FileUtils.openOutputStream(file);
            workbook.write(outputStream);
            outputStream.close();
            workbook.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * 从Excel文件中，读取内容
     * @param filePath
     * @return 类型是List<String[]>，list大小表示行的大小，每个数组的大小表示列的大小
     */
    public static List<String[]> readExcel(String filePath) {
        List<String[]> dataList = new ArrayList<String[]>();
        try {
            // >> 从输入流创建工作簿
            XSSFWorkbook workbook = new XSSFWorkbook(FileUtils.openInputStream(new File(filePath)));
            // >> 获取第0个工作表
            XSSFSheet sheet = workbook.getSheetAt(0);
            // >> 获取数据
            // 获取最后一行行号
            int lastRowNum = sheet.getLastRowNum();
            // lastRowNum + 1，因为lastRowNum表示当前行是最后一行，但是下标从0开始的
            for (int i = 0; i < lastRowNum + 1; i++) {
                // 获取第i行
                XSSFRow row = sheet.getRow(i);
                // 获取最后一个单元格的列号
                short lastCellNum = row.getLastCellNum();
                // 存储一行的数据
                String[] rowValue = new String[lastCellNum];
                for (int j = 0; j < lastCellNum; j++) {
                    // 获取第i行的第j列的单元格
                    XSSFCell cell = row.getCell(j);
                    // 由于前面放进去的值都是String，故这里以String的形式获取
                    String cellValue = cell.getStringCellValue();
                    rowValue[j] = cellValue;
```

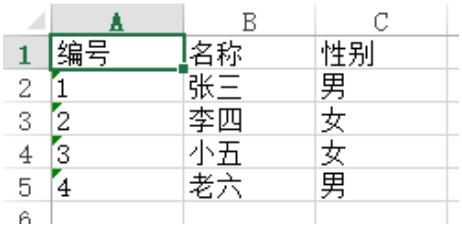
```
                }
                dataList.add(rowValue);
            }
            // >> 关闭
            workbook.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return dataList;
    }

}
```

创建 Excel 文件

执行 testCreateExcel()测试方法，创建 Excel 文件



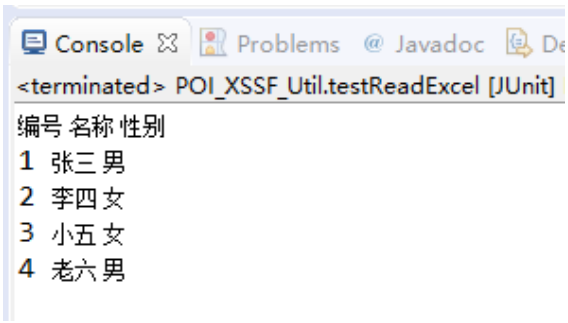
内容



	A	B	C
1	编号	名称	性别
2	1	张三	男
3	2	李四	女
4	3	小五	女
5	4	老六	男
6			

读取 Excel 文件

执行 testReadExcel()测试方法，读取 Excel 文件，输出结果如下



结束日期：2018-08-02