案例：<https://gitee.com/zx19890628/spring-boot-example/tree/master/lab_057_util_excel>

10w行级别数据的Excel导入优化记录：

<https://mp.weixin.qq.com/s/QQ-OFZ8laQHbaIOOpBZNzQ>

<http://www.cnblogs.com/keatsCoder/p/13217561.html>

# POI

一、导入依赖

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.14</version>

</dependency>

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.14</version>

</dependency>

二、编写数据采集代码(其中需要编写的代码是：1.查询的数据；2.表头；3.封装数据)

//1.得到查询的结果

UserDao ud = new UserDao();

List<User> li = new ArrayList<User>();

li = ud.getAll();

//2.设置导出excel表头

String[] titles = {"姓名","性别","年龄","年级"};

//将表头存储起来

ArrayList<String> fileName=new ArrayList<String>();

for (int i = 0; i < titles.length; i++) {

fileName.add(titles[i]); //向Excel表中插入表头

}

//3.创建一个集合，用来存放查询出来的每一条数据（数组类型，每个字段信息为数组的一个元素）

ArrayList<String[]> list=new ArrayList<String[]>(); //返回的list存放着一个个的数组：每个数组存入一条记录

for (User user : li) {

//将每一条数据生成一个字符串数组，存放在集合中

String[] yuan = {user.getName(),user.getGender(),user.getAge()+"",user.getGrade()};

list.add(yuan); //list里存放着一个个的数组：每个数组存入一条记录

}

//4.取出数组中的每一个元素，存到集合中

ArrayList filedData=new ArrayList();

for (int i = 0; i < list.size(); i++) { //第一重循环：遍历list,一个数组一个数组地遍历(一个数组就是一条记录)

ArrayList<String> datalist=new ArrayList<String>();

for (int j = 0; j < titles.length; j++) { //第二重循环：遍历list里的数组[ ]

String data=list.get(i)[j];

datalist.add(data); //每个数组的每个元素取完，放到一个集合(datalist)中

}

filedData.add(datalist); //每个数组取完，整条记录的数据放到另一个集合(filedData)中

}

//5.向服务器的响应中添加一个输出流

OutputStream out = response.getOutputStream();

//6.清空输出流

response.reset();

//7.设置导出Excel报表的导出形式

response.setContentType("application/vnd.ms-excel");

ExcelFileGenerator generator = new ExcelFileGenerator(fileName, filedData);

try {

generator.expordExcel(out);//导出

System.setOut(new PrintStream(out));

out.flush();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

} finally{

if (out!=null) {

out.close();

}

}

# Jxl

类型POI的工具

# easyExcel

GitHub 源码地址

<https://github.com/weiwosuoai/spring-boot-tutorial/tree/master/spring-boot-excel>

参考文献

<https://github.com/alibaba/easyexcel>

Apache poi、jxl 的缺陷

在说如何实现之前，我们先来讨论一下传统 Excel 框架的不足！除了上面说的，

Apache poi、jxl 都存在生成 excel 文件不够简单优雅快速外，它们都还存在一个严重的问题，那就是非常耗内存，严重时会导致内存溢出。

POI 虽然目前来说，是 excel 解析框架中被使用最广泛的，但这个框架并不完美。

为什么这么说呢？

开发者们大部分使用 POI，都是使用其 userModel 模式。而 userModel 的好处是上手容易使用简单，随便拷贝个代码跑一下，剩

下就是写业务转换了，虽然转换也要写上百行代码，但是还是可控的。

然而 userModel 模式最大的问题是在于，对内存消耗非常大，一个几兆的文件解析甚至要用掉上百兆的内存。

现实情况是，很多应用现在都在采用这种模式，之所以还正常在跑是因为并发不大，并发上来后，一定会OOM或者频繁的 full gc。

EasyExcel 解决了什么

主要来说，有以下几点：

传统 Excel 框架，如 Apache poi、jxl 都存在内存溢出的问题；

传统 excel 开源框架使用复杂、繁琐;

EasyExcel 底层还是使用了 poi, 但是做了很多优化，如修复了并发情况下的一些 bug, 具体修复细节，可阅读官方文档https://github.com/alibaba/easyexcel；

1. 添加依赖

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>easyexcel</artifactId>

<version>1.1.2-beta5</version>

</dependency>

2. 模型类

@Data

@NoArgsConstructor

@AllArgsConstructor

@ToString

@Builder

public class ReadModel extends BaseRowModel {

@ExcelProperty(value = "姓名", index = 0)

private String name;

@ExcelProperty(value = "出生日期", index = 1)

private Date brith;

@ExcelProperty(value = "地址", index = 2)

private String address;

@ExcelProperty(value = "email", index = 3)

private String email;

}

3. 测试

public static void main(String[] args) throws Exception {

List<WriteModel> wl = new ArrayList<>();

for (int i = 0; i < 100; i++) {

WriteModel model = WriteModel.builder().name("测试人员").brith(new Date()).address("6666").email("11@qq.com").build();

wl.add(model);

}

// 创建一个Excel

OutputStream os = new FileOutputStream("src/main/resources/excel/write.xlsx");

ExcelWriter writer = EasyExcelFactory.getWriter(os);

//写第一个sheet, sheet1 数据全是List<String> 无模型映射关系

Sheet sheet1 = new Sheet(1, 0, WriteModel.class);

sheet1.setSheetName("第一个");

//设置列宽 设置每列的宽度

// Map columnWidth = new HashMap();

// columnWidth.put(0,10000);

// columnWidth.put(1,40000);

// columnWidth.put(2,10000);

// columnWidth.put(3,10000);

// sheet1.setColumnWidthMap(columnWidth);

//or 设置自适应宽度

sheet1.setAutoWidth(Boolean.TRUE);

writer.write(wl, sheet1);

writer.finish();

os.close();

}