参考博客：<https://www.cnblogs.com/linjiqin/p/10975761.html>

<https://blog.csdn.net/weixin_40826349/article/details/88995812>

<https://www.cnblogs.com/dinghaoran/p/12462671.html>

ali：easyexcel

阿里的Excel导入导出：

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

<https://www.yuque.com/easyexcel/doc/api>

## 导入

依赖

|  |
| --- |
| <**dependency**>  <**groupId**>com.alibaba</**groupId**>  <**artifactId**>easyexcel</**artifactId**>  <**version**>2.2.6</**version**> </**dependency**> |

数据读取监听：

这里是一个通用的监听方式写法，适用多地方使用，小数据读取，如果是大数据导入，最好采用文档里的方法进行操作，一个Excel一个监听处理类：

他可以规定多少条数据进行数据库操作一次，然后清空内存数据，减轻服务器压力。

|  |
| --- |
| // 有个很重要的点 DemoDataListener 不能被spring管理，要每次读取excel都要new,然后里面用到spring可以构造方法传进去  public class DemoDataListener extends AnalysisEventListener<DemoData> {  private static final Logger LOGGER = LoggerFactory.getLogger(DemoDataListener.class);  /\*\*  \* 每隔5条存储数据库，实际使用中可以3000条，然后清理list ，方便内存回收  \*/  private static final int BATCH\_COUNT = 5;  List<DemoData> list = new ArrayList<DemoData>();  /\*\*  \* 假设这个是一个DAO，当然有业务逻辑这个也可以是一个service。当然如果不用存储这个对象没用。  \*/  private DemoDAO demoDAO;  public DemoDataListener() {  // 这里是demo，所以随便new一个。实际使用如果到了spring,请使用下面的有参构造函数  demoDAO = new DemoDAO();  }  /\*\*  \* 如果使用了spring,请使用这个构造方法。每次创建Listener的时候需要把spring管理的类传进来  \*  \* @param demoDAO  \*/  public DemoDataListener(DemoDAO demoDAO) {  this.demoDAO = demoDAO;  }  /\*\*  \* 这个每一条数据解析都会来调用  \*  \* @param data  \* one row value. Is is same as {@link AnalysisContext#readRowHolder()}  \* @param context  \*/  @Override  public void invoke(DemoData data, AnalysisContext context) {  LOGGER.info("解析到一条数据:{}", JSON.toJSONString(data));  list.add(data);  // 达到BATCH\_COUNT了，需要去存储一次数据库，防止数据几万条数据在内存，容易OOM  if (list.size() >= BATCH\_COUNT) {  saveData();  // 存储完成清理 list  list.clear();  }  }  /\*\*  \* 所有数据解析完成了 都会来调用  \*  \* @param context  \*/  @Override  public void doAfterAllAnalysed(AnalysisContext context) {  // 这里也要保存数据，确保最后遗留的数据也存储到数据库  saveData();  LOGGER.info("所有数据解析完成！");  }  /\*\*  \* 加上存储数据库  \*/  private void saveData() {  LOGGER.info("{}条数据，开始存储数据库！", list.size());  demoDAO.save(list);  LOGGER.info("存储数据库成功！");  }  } |

|  |
| --- |
| **package** com.bkgtsoft.alu.utils;  **import** com.alibaba.excel.context.AnalysisContext; **import** com.alibaba.excel.event.AnalysisEventListener; **import** lombok.Data; **import** lombok.extern.log4j.Log4j2;  **import** java.util.ArrayList; **import** java.util.List;  */\*\*  \** ***@author*** *iszengziqiang@163.com  \** ***@date*** *2020/7/14 11:07  \** ***@Version:*** *1.0  \** ***@desc*** *//****todo*** *\*/* @Log4j2 @Data **public class** ExcelListener<T> **extends** AnalysisEventListener {   **private** List<T> **datas** = **new** ArrayList();   @Override  **public void** invoke(Object o, AnalysisContext analysisContext) {  ***log***.info(**"每一条数据：{}"**, o.toString());  **datas**.add((T) o);  }   @Override  **public void** doAfterAllAnalysed(AnalysisContext analysisContext) {   } } |

Excel与实体类对应：ID或表头名都可以

|  |
| --- |
| **package** com.bkgtsoft.alu.entity.po;  **import** com.alibaba.excel.annotation.ExcelProperty; **import** com.alibaba.excel.metadata.BaseRowModel; **import** com.baomidou.mybatisplus.annotation.IdType; **import** com.baomidou.mybatisplus.annotation.TableId; **import** io.swagger.annotations.ApiModel; **import** io.swagger.annotations.ApiModelProperty; **import** lombok.Data; **import** lombok.EqualsAndHashCode; **import** lombok.experimental.Accessors;  **import** java.io.Serializable; **import** java.time.LocalDateTime;  */\*\*  \* <p>  \*   \* </p>  \*  \** ***@author*** *zzq  \** ***@since*** *2020-07-14  \*/* @Data **public class** EqptTeaching **extends** BaseRowModel {   **private static final long *serialVersionUID*** = 1L;   @TableId(value = **"id"**, type = IdType.***AUTO***)  **private** Integer **id**;   @ApiModelProperty(value = **"排课号"**)  @ExcelProperty(value = **"排课号"**, index = 0)  **private** String **courseCode**;   @ApiModelProperty(value = **"课程名称"**)  @ExcelProperty(value = **"课程名称"**, index = 1)  **private** String **courseName**;   @ApiModelProperty(value = **"组别"**)  @ExcelProperty(value = **"组别"**, index = 2)  **private** String **tranches**;   @ApiModelProperty(value = **"主讲教员"**)  @ExcelProperty(value = **"主讲教员"**, index = 3)  **private** String **lecturer**;   @ApiModelProperty(value = **"主讲教员ID"**)  **private** Integer **lecturerId**;   @ApiModelProperty(value = **"总人数"**)  @ExcelProperty(value = **"总人数"**, index = 4)  **private** Integer **totalCount**;   @ApiModelProperty(value = **"班级人数"**)  @ExcelProperty(value = **"班级人数"**, index = 5)  **private** String **classCount**;   @ApiModelProperty(value = **"学时"**)  @ExcelProperty(value = **"学时"**, index = 6)  **private** String **period**;   @ApiModelProperty(value = **"学分"**)  @ExcelProperty(value = **"学分"**, index = 7)  **private** String **credit**;   @ApiModelProperty(value = **"修习类型"**)  @ExcelProperty(value = **"修习类型"**, index = 8)  **private** String **learnType**;   @ApiModelProperty(value = **"承担单位"**)  @ExcelProperty(value = **"承担单位"**, index = 9)  **private** String **assumeDepa**;   @ApiModelProperty(value = **"教研室"**)  @ExcelProperty(value = **"教研室"**, index = 10)  **private** String **classroom**;   @ApiModelProperty(value = **"考核方式"**)  @ExcelProperty(value = **"考核方式"**, index = 11)  **private** String **evaluationMode**;   @ApiModelProperty(value = **"层次"**)  @ExcelProperty(value = **"排课号"**, index = 12)  **private** String **gradation**;   @ApiModelProperty(value = **"备注"**)  @ExcelProperty(value = **"备注"**, index = 13)  **private** String **remark**;   **private** LocalDateTime **createTime**;   **private** LocalDateTime **updateTime**;   } |

读取数据：可以采用同步返回的方式

|  |
| --- |
| @ApiOperation(**"导入Excel"**) @PostMapping(**"v1/import"**) **public** RespBean<List<EqptTeaching>> importExcel(@RequestParam(**"file"**) MultipartFile file) **throws** IOException {  ExcelListener<EqptTeaching> excelListener = **new** ExcelListener();  EasyExcel.*read*(file.getInputStream(), EqptTeaching.**class**, excelListener).sheet(0).doRead();  List<EqptTeaching> datas = excelListener.getDatas();  **return** RespBean.*succeed*(GeneralRespEnum.***QUERY\_SUCCESS***, datas); } |

同步读数据的方式：

|  |
| --- |
| /\*\*  \* 同步的返回，不推荐使用，如果数据量大会把数据放到内存里面  \*/  @Test  public void synchronousRead() {  String fileName = TestFileUtil.getPath() + "demo" + File.separator + "demo.xlsx";  // 这里 需要指定读用哪个class去读，然后读取第一个sheet 同步读取会自动finish  List<DemoData> list = EasyExcel.read(fileName).head(DemoData.class).sheet().doReadSync();  for (DemoData data : list) {  LOGGER.info("读取到数据:{}", JSON.toJSONString(data));  }  // 这里 也可以不指定class，返回一个list，然后读取第一个sheet 同步读取会自动finish  List<Map<Integer, String>> listMap = EasyExcel.read(fileName).sheet().doReadSync();  for (Map<Integer, String> data : listMap) {  // 返回每条数据的键值对 表示所在的列 和所在列的值  LOGGER.info("读取到数据:{}", JSON.toJSONString(data));  }  } |

然后就可以进行数据的处理了，保存数据库或返回给前端展示

## 导出

将选择的数据导出为Excel表格到桌面：

封装成一个list列表进行操作即可。

|  |
| --- |
| @ApiOperation(**"导出excel"**) @GetMapping(**"v1/export"**) **public void** createExcel(HttpServletRequest request, HttpServletResponse response) **throws** IOException {  **try** {  *//从数据库查询的记录* List<EqptTeaching> datas = **new** ArrayList<EqptTeaching>();  **for** (**int** i = 0; i < 5; i++) {  EqptTeaching dataDto = **new** EqptTeaching();  dataDto.setId(i + 1);  dataDto.setCourseName(**"语文"** + i);  dataDto.setClassCount(**"60"** + i);  datas.add(dataDto);  }  String exportFileName = **"计算机学院飞扬班"**;  *// 这里注意使用swagger测试会有问题，请直接用浏览器或者用postman* response.setContentType(**"application/vnd.ms-excel"**);  response.setCharacterEncoding(**"utf-8"**);  *// 这里URLEncoder.encode可以防止中文乱码* String fileName = URLEncoder.*encode*(exportFileName, **"UTF-8"**);  response.setHeader(**"Content-disposition"**, **"attachment;filename="** + fileName + **".xlsx"**);  EasyExcel.*write*(response.getOutputStream(), EqptTeaching.**class**).sheet(**"模板"**).doWrite(datas);  } **catch** (Exception e) {  ***log***.error(**"创建excel异常"**, e.getMessage());  response.reset();  response.setContentType(**"application/json"**);  response.setCharacterEncoding(**"utf-8"**);  RespBean<Object> failed = RespBean.*failed*(-1, **"下载文件失败："** + e.getMessage());  response.getWriter().println(JSONObject.*toJSONString*(failed));  } } |