引入依赖

<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>easyexcel</artifactId>  
 <version>2.1.1</version>  
 </dependency>

往excel表格里面写数据（下载excel）

1、建实体类（对应excel表格里面的字段）

@Data  
public class UserData {  
 @ExcelProperty(value="用户编号", index=0)

@ColumnWidth(10)  
 private Integer uid;

@ExcelProperty(value="用户名称", index=1)

@ColumnWidth(20)  
 private String username;

@ExcelProperty(value=”创建时间”, index=2)

@DateTimeFormat(“yyyy-MM-dd HH:mm:ss”)

private Date createTime;

}

1. 写代码

// 下载文件  
@GetMapping("download")  
public void downLoad(HttpServletResponse response) throws IOException {  
 // 设置utf编码， 防止乱码  
 response.setContentType("application/vnd.ms-excel");  
 response.setCharacterEncoding("utf-8");  
  
 // 设置下载的文件名字  
 String fileName = URLEncoder.encode("测试", "UTF-8");  
 response.setHeader("Content-Disposition", "attachment; filename\*=UTF-8''"+fileName+".xlsx");  
  
 // 输出流  
 ServletOutputStream outputStream = response.getOutputStream();  
  
 // 查询数据库  
 List<TestExcelEntity> list = testExcelService.selectList();  
  
 // 用easyExcel将数据写入excel  
 EasyExcel.write(outputStream, TestExcelEntity.class).sheet("测试sheet").doWrite(list);  
}

读excel表格里面的数据存到数据库

1. 创建实体类，用于接受excel表格里面的字段

@Data  
public class UserData {  
 @ExcelProperty(value="用户编号", index=0) // 对应excel表格的字段

@ColumnWidth(20) // excel表格的宽度

@ExcelIgnore // 忽略此字段

@NumberFormat // 格式化数字

private Integer uid;

@ExcelProperty(value="用户名称", index=1)

@ColumnWidth(20)  
 private String username;

@ExcelProperty(value=”出生日期” , index=2)

@DateTimeFormat(“yyyy-MM-dd HH:mm:ss”) // 格式化日期

Private Date time;  
}

1. Controller层

@PostMapping("/uploadFile")  
public String uploadFile(MultipartFile file) {  
 uploadFileService.uploadFile(file);  
 return "上传成功";  
}

1. Service层

@Override  
public void uploadFile(MultipartFile file) {  
 try {  
 EasyExcel.read( file.getInputStream(), UserEntity.class, new ExcelListen(uploadFileDao) ).sheet().headRowNumber(1).doRead();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

EasyExcel.read() 方法里面有三个参数：

1）接收到前端传过来的文件的输入流

2）实体类（开始创建的实体类，用于接受excel表格的字段）的class

3）实例化监听器，一行一行的读excel表格

1. 监听器

public class ExcelListen extends AnalysisEventListener<UserEntity> {  
  
 private UploadFileDao uploadFileDao;  
  
 public ExcelListen(UploadFileDao uploadFileDao) {  
 this.uploadFileDao = uploadFileDao;  
 }  
  
  
 @Override  
 public void invoke(UserEntity userEntity, AnalysisContext analysisContext) {  
  
 uploadFileDao.insertMap(userEntity); // 调用dao层将数据存到数据库  
 }  
  
 @Override  
 public void doAfterAllAnalysed(AnalysisContext analysisContext) {  
  
 }

// 处理异常

@Override

public void onException(Exception exception, AnalysisContext context) {  
 System.err.println(exception.getMessage());  
 // 如果是某一个单元格的转换异常 能获取到具体行号  
 // 如果要获取头的信息 配合invokeHeadMap使用  
 if (exception instanceof ExcelDataConvertException) {  
 ExcelDataConvertException excelDataConvertException = (ExcelDataConvertException)exception;  
 System.err.println("第" + (excelDataConvertException.getRowIndex()+1) + "行，第" + (excelDataConvertException.getColumnIndex()+1) + "列有异常");  
 }  
 }

}