SpringBoot+EasyExcel+Mybatis-plus优雅实现通用Excel数据批量导入，快用起来吧！
==========

Hi,大家好，我是抢老婆酸奶的小肥仔。

在我们日常开发中，有个功能几乎是没办法绕开的，那就是Excel数据的导入。当然也有很多工具支持导入导出,比如：Apache POI，jxl，由于Apache POI在加载大量数据时会出现OOM，因此阿里在其基础上进行了封装形成了EasyExcel，这便是我们今天要说的主角。

1、EasyExcel简介
==========

官网：[easyexcel.opensource.alibaba.com/](http://cxyroad.com/"https://easyexcel.opensource.alibaba.com/")

按照官网介绍，EasyExcel具有以下特点：

1. 基于java实现的
2. 快捷、简洁、解决大文件内存溢出（即OOM）
3. 不用考虑性能、内存等因素情况下，快速完成Excel的读、写等功能。

从简介中，知道使用EasyExcel操作Excel时，我们只需要业务本身即可。

2、程序实现
======

上述我们稍稍简介了下EasyExcel，接下来我们来看看怎样使用。

引入依赖

```
<dependency>
```

```
  <groupId>com.alibaba</groupId>
  <artifactId>easyexcel</artifactId>
  <version>3.3.2</version>
</dependency>
```

2.1 读取Excel

按照官网给出的例子，一共有4种读取Excel写法，通过官网的例子来看，其实所有例子都是要以`ReadListener<T>` 为基础。我们以官网提供的方法一（PageReadListener）为例来说下EasyExcel读取的实现。

```
EasyExcel.read(fileName, DemoData.class, new
PageReadListener<DemoData>(dataList -> {
    for (DemoData demoData : dataList) {
        log.info("读取到一条数据{}", JSON.toJSONString(demoData));
    }
}),1000).sheet().doRead();
```

这种方法直接使用了PageReadListener监听器，我们来看看PageReadListener源码：

```
public class PageReadListener<T> implements ReadListener<T> {
    public static int BATCH_COUNT = 100;
    private List<T> cachedDataList;
    private final Consumer<List<T>> consumer;
    private final int batchCount;

    public PageReadListener(Consumer<List<T>> consumer) {
        this(consumer, BATCH_COUNT);
    }

    public PageReadListener(Consumer<List<T>> consumer, int
batchCount) {
        this.cachedDataList =
ListUtils.newArrayListWithExpectedSize(BATCH_COUNT);
        this.consumer = consumer;
```

```
        this.batchCount = batchCount;
    }

    public void invoke(T data, AnalysisContext context) {
        this.cachedDataList.add(data);
        if (this.cachedDataList.size() >= this.batchCount) {
            this.consumer.accept(this.cachedDataList);
            this.cachedDataList =
ListUtils.newArrayListWithExpectedSize(this.batchCount);
        }

    }

    public void doAfterAllAnalysed(AnalysisContext context) {
        if (CollectionUtils.isNotEmpty(this.cachedDataList)) {
            this.consumer.accept(this.cachedDataList);
        }

    }
}
```

在PageReadListener中给了两个构造方法，两者唯一区别就是一个采用了默认的读取条数，一个采用了用户自定义的条数。

在源码中还有两个方法：`invoke`和`doAfterAllAnalysed`，他们都是实现ReadListener接口里面定义的方法。

> **`invoke`**：表示每解析完一条数据就会调用该初始方法，因此很多条件筛选或业务我们可以放在里面实现。
>
>
> **`doAfterAllAnalysed`**：表示每解析完一个sheet页后调用该方法。

从源码中知道，invoke方法中当集合的长度大于等于设置的长度时，则执Consumer，执行完成后在进行初始化集合；doAfterAllAnalysed方法中在获取完数据后，判断当前集合是否还有数据，有的话则执行Consumer。

2.2 整合Mybutis-plus

我们通过方法一了解了PageReadListener的实现过程，那么我们也可以仿照PageReadListener、结合Mybatis-plus来实现一个通用批量导入的功能。

我们来简单引入Mybatis-plus。

1、引入依赖

```
<dependency>
  <groupId>com.baomidou</groupId>
  <artifactId>mybatis-plus-boot-starter</artifactId>
  <version>3.5.3</version>
</dependency>
```

2、Mybatis-plus配置

```
/**
 * @author: jiangjs
 * @description: mybatis-plus配置
 * @date: 2023/10/20 15:16
 **/
@Configuration
@MapperScan(value = "com.**.mapper.**")
public class MybatisPlusConfig {
}
```

3、数据库开启批量插入

在Mysql批量插入数据时，我们一般采用insert ino table (xxx,xxxx) valus (11,22),(33,44)，这种方式减少了数据库的连接，提高插入效率，而mybatis这样执行批处理需要在数据库url配置上添加`rewriteBatchedStatements=true`，进行批处理开启。

```
```

```
url: jdbc:mysql://127.0.0.1:3308/xxxxx?characterEncoding=utf-8&allowMultiQueries=true&zeroDateTimeBehavior=convertToNull&useSSL=false&useUnicode=true&rewriteBatchedStatements=true
```

上述准备工作做好后，我们开始编写代码。

2.2.1 通用监听器

```
/**
 * @author: jiangjs
 * @description: 公用数据分析监听器
 * @date: 2023/12/26 14:30
 **/
@Slf4j
public class PublicReadExcelListener<M extends BaseMapper<T>,T> implements ReadListener<T> {

    private final static int BATCH_NUM = 1000;
    private final int batchNum;
    List<T> data = new ArrayList<>();

    private final ServiceImpl<M,T> service;

   public PublicReadExcelListener(ServiceImpl<M,T> service){
        this(service,BATCH_NUM);
    }

    public PublicReadExcelListener(ServiceImpl<M,T> service,int batchNum){
        this.service = service;
        this.batchNum = batchNum;
    }
     /**
     * 每解析一行数据就调用该方法
     * @param t 实体
     * @param analysisContext 分析上下文
     */
    @Override
    public void invoke(T t, AnalysisContext analysisContext) {
        data.add(t);
        if (data.size() >= batchNum){
            service.saveBatch(data,batchNum);
```

```
            data.clear();
        }
    }

    /**
     * 每解析完一页sheet页就会调用该方法
     * @param analysisContext 分析上下文
     */
    @SneakyThrows
    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {
        log.info("读取当前文件第【
"+(analysisContext.readSheetHolder().getSheetNo() + 1)+"】sheet页");
        if (data.size() > 0){
            service.saveBatch(data,batchNum);
        }
        data = new ArrayList<>();
    }
}
```

编写的通用监听器中，我们会发现在类`PublicReadExcelListener<M extends BaseMapper<T>,T>` ，上仿造了Mybatis-plus的通用`ServiceImpl<M extends BaseMapper<T>, T>` ，其主要目的是为了在初始化时赋值`ServiceImpl<M,T> service`，从而调用mybatis-plus中的`saveBatch(Collection<T> entityList, int batchSize)` 方法，实现批量插入。

上述通用监听器与PageReadListener逻辑基本相似，唯一不同的是实现了Mybatis-plus中的saveBatch而已。

2.2.2 测试批量插入

上述实现了一个通用监听器，且整合了Mybatis-plus中的ServiceImpl，那么我们来测试一下，看看效果。

由于使用了ServiceImpl，因此我们在Service中也要继承这个父类。

```
@Service
@Slf4j
public class UserServiceImpl extends ServiceImpl<UserMapper, User>
```

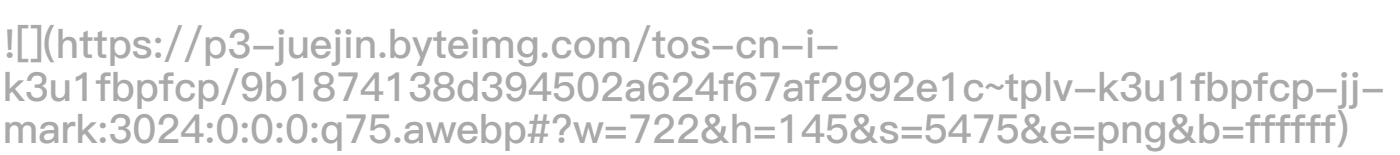
```
implements UserService{
    private final static List<String> EXCEL_SUFFIX =
Arrays.asList("xls","xlsx");
    @Override
    public void importDataByExcel(MultipartFile file) {
        String filename = file.getOriginalFilename();
        Assert.notNull(filename,"文件名称不能为空");
        String suffix = filename.substring(filename.lastIndexOf(".")+1);
        if (!EXCEL_SUFFIX.contains(suffix)){
            throw new RuntimeException("上传的文件不是Excel文件!");
        }
         try (InputStream ism = file.getInputStream()){
            EasyExcel.read(ism,User.class,new
PublicReadExcelListener<>(this,1000)).sheet(0).headRowNumber(1).doRe
ad();
        }catch (Exception ex){
            log.error("导入文件【"+filename+"】报错：",ex);
            ex.printStackTrace();
        }
    }
}

```

User实体中只有两个字段id和userName。

```
/**
 * @author: jiangjs
 * @description:
 * @date: 2023/12/26 15:35
 **/
@Data
@TableName("t_user")
public class User {
    @TableId(type = IdType.AUTO)
    private Integer id;

    @ExcelProperty("用户名")
    private String userName;
}

```

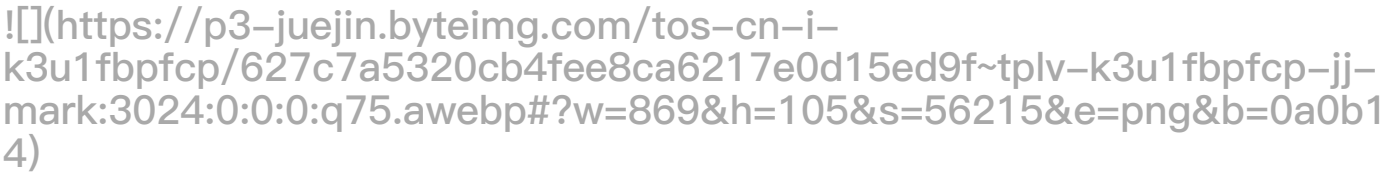
`@ExcelProperty`：EasyExcel提供的注解，标志当前字段的名称，用于读取数据时表头与当前字段进行匹配。

我们准备一个Excel表格里面只有一个用户名字段，如图：



运行测试结果：

数据库数据



从测试结果来说，数据已经正常导入。

2.2.3 筛选条件

上述初步实现了导入，但是平时的业务需求不可能只是单纯的数据导入，往往会对Excel表格中的数据进行过滤筛选，把符合要求的数据导入到库中。So Easy，我们也来实现下通用条件的筛选功能。

在函数式编程中，Function<T,R>是传入T，返回R，那么我们就可以基于Function实现多条件过滤。

来改造一下通用监听器，先贴代码

```
```

```
/**
 * @author: jiangjs
 * @description: 公用数据分析监听器
 * @date: 2023/12/26 14:30
 **/
@Slf4j
public class PublicReadExcelListener<M extends BaseMapper<T>,T>
implements ReadListener<T> {

    private final static int BATCH_NUM = 1000;

    private final int batchNum;

    List<T> data = new ArrayList<>();

    private final ServiceImpl<M,T> service;
    private Function<T,Boolean>[] conditions = null;

    public PublicReadExcelListener(ServiceImpl<M,T> service){
        this(service,BATCH_NUM);
    }
    public PublicReadExcelListener(ServiceImpl<M,T> service,int
batchNum){
        this.service = service;
        this.batchNum = batchNum;
    }

    @SafeVarargs
    public PublicReadExcelListener(ServiceImpl<M,T> service,
Function<T,Boolean>... conditions){
        this(service,BATCH_NUM);
        this.conditions = conditions;
    }

    @SafeVarargs
    public PublicReadExcelListener(ServiceImpl<M,T> service, int
batchNum,Function<T,Boolean>... conditions){
        this(service,batchNum);
        this.conditions = conditions;
    }

    @Override
    public void invoke(T t, AnalysisContext analysisContext) {
        //解析数据，添加到集合中
        Boolean conform = Objects.nonNull(conditions) ?
this.moreConditions(t,conditions) : Boolean.TRUE;
        if (conform) {
            data.add(t);
```

```
        }
        if (data.size() >= batchNum){
            service.saveBatch(data,batchNum);
            data.clear();
        }
    }

    /**
     * 每解析完一页sheet页就会调用该方法
     * @param analysisContext 分析上下文
     */
    @SneakyThrows
    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {
        log.info("读取当前文件第【
"+(analysisContext.readSheetHolder().getSheetNo() + 1)+"】sheet页");
        if (data.size() > 0){
            service.saveBatch(data,batchNum);
        }
        data = new ArrayList<>();
    }

    /**
     * 多条件过滤
     * @param t 实体
     * @param conditions 条件
     * @return 返回当前数据是否符合条件
     */
    @SafeVarargs
    private final Boolean moreConditions(T t, Function<T, Boolean>...
conditions){
        Boolean conform = Boolean.TRUE;
        for (Function<T, Boolean> func : conditions) {
            conform = func.apply(t) && conform;
        }
        return conform;
    }
}

```

在原来的通用监听器上，我们添加了一个Function的数组：`private Function<T,Boolean>[] conditions = null;` 主要是用于接收各筛选条件。在invoke方法中，判断是否有筛选条件，有就执行方法`moreConditions(T t, Function<T, Boolean>... conditions)` ，进行条件筛选。

在moreConditions方法中是将当前实体中的各条件执行结果进行与运算，得到

最终的结果是否符合条件。

2.2.4 测试筛选条件

准备一个Excel，只有两条数据，如：李四、王五。如图：

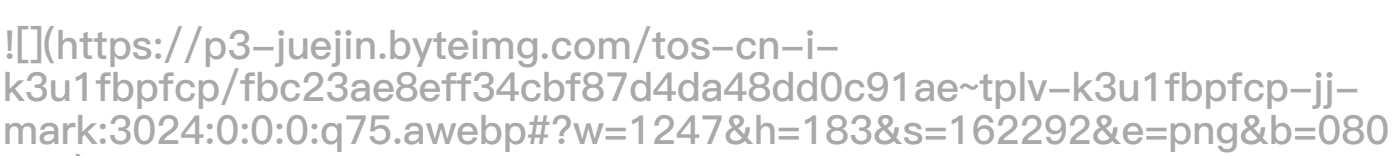
需求：导入王五。

```
/**
 * @author: jiangjs
 * @description:
 * @date: 2023/12/26 15:38
 **/
@Service
@Slf4j
public class UserServiceImpl extends ServiceImpl<UserMapper, User>
implements UserService{

    private final static List<String> EXCEL_SUFFIX =
Arrays.asList("xls","xlsx");

    @Override
    public void importDataByExcel(MultipartFile file) {
        String filename = file.getOriginalFilename();
        Assert.notNull(filename,"文件名称不能为空");
        String suffix = filename.substring(filename.lastIndexOf(".")+1);
        if (!EXCEL_SUFFIX.contains(suffix)){
            throw new RuntimeException("上传的文件不是Excel文件!");
        }
         try (InputStream ism = file.getInputStream()){
            EasyExcel.read(ism,User.class,new
PublicReadExcelListener<>(this,1000,
                    t -> Objects.equals(t.getUserName(),"王五")))
                    .sheet(0).headRowNumber(1).doRead();
        }catch (Exception ex){
            log.error("导入文件【"+filename+"】报错：",ex);
            ex.printStackTrace();
```

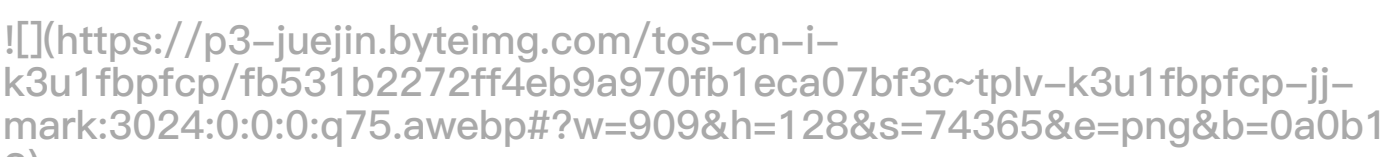
```
        }
    }
}
```

> 注：初始化 PublicReadExcelListener(ServiceImpl<M,T> service, int batchNum,Function<T,Boolean>... conditions)时多条件写入，例如：new PublicReadExcelListener<>(this,t -> Objects.equals(t.getUserName(),"王五"),t -> t.getUserName().contains("王"))

运行测试结果：



数据库结果：



执行结果中可以看到是符合需求的。

2.2.5 多sheet页、多行head导入

EasyExcel中readSheet()是可以指定需要读取哪个sheet页的，默认情况下值是0，也就是读取第一行，headRowNumber()可以指定从第几行表头进行开始读取数据。那么我们是不是封装一个方法提供这些参数，就可以进行多sheet页，多表头读取数据呢？那肯定是必须的必了。

直接开干吧。

字段实体类：

```
/**
 * @author: jiangjs
 * @description: excel导入实体
 * @date: 2023/12/26 16:55
 **/
@Data
@Accessors(chain = true)
public class ExcelImportParamVO {
    /**
     * 文件
     */
    private MultipartFile file;

    /**
     * 实体
     */
    private Class<?> clazz;

    /**
     * 读取监听器
     */
    private ReadListener<?> readListener;

    /**
     * 自定义格式转换
     */
    private List<Converter<?>> converters;

    /**
     * 文件sheet页数量
     */
    private Integer sheetNum = 1;
    /**
     * 每个sheet页头部行数，按照sheet顺序进行赋值
     */
    private Integer[] headNums = {1};

}
```

如果只需要导入一个sheet页且表头也是从第一行开始时，则不需要赋值sheetNum和headNums。

EasyExcel工具类：

```
/**
 * @author: jiangjs
 * @description:
 * @date: 2023/12/26 14:19
 **/
@Slf4j
public class EasyExcelUtil {

    /**
     * 导入Excel文件(单、多sheet页)
     * @param param 文件读取实体
     */
    public static void importExcelFile(ExcelImportParamVO param){
        if (Objects.isNull(param.getFile())){
            throw new RuntimeException("请上传Excel文件");
        }
        try (InputStream ism = param.getFile().getInputStream()){
            ExcelReader reader =
EasyExcel.read(ism,param.getClazz(),param.getReadListener()).build();
            List<ReadSheet> sheets = new ArrayList<>();
            for (int i = 0; i < param.getSheetNum(); i++) {
                Integer headNum = param.getHeadNums().length < i ? 1 :
param.getHeadNums()[i];
                ExcelReaderSheetBuilder builder =
EasyExcel.readSheet(i).headRowNumber(headNum);
                if (CollectionUtils.isNotEmpty(param.getConverters())){
                    for (Converter<?> converter : param.getConverters()) {
                        builder.registerConverter(converter);
                    }
                }
                sheets.add(builder.build());
            }
            reader.read(sheets);
            reader.finish();
        }catch (Exception ex){
            log.error("导入文件【"+param.getFile().getOriginalFilename()+"】
报错：",ex);
            ex.printStackTrace();
        }
    }
}

```

工具类中只是同步实现了多sheet页，有兴趣的小伙伴可以试试异步实现多sheet页导入。

当前工具类主要执行顺序：

1. 首先通过`ExcelReader reader = EasyExcel.read(ism,param.getClazz(),param.getReadListener()).build();`创建ExcelReader
2. 创建集合`List<ReadSheet> sheets = new ArrayList<>();`用于存放需要执行的sheet页
3. 根据文件流，指定sheet页和当前页的标题，创建ExcelReaderSheetBuilder。`ExcelReaderSheetBuilder builder = EasyExcel.readSheet(i).headRowNumber(headNum);`
4. 加入自定义单元格格式转换，创建ReadSheet，并添加到集合中。
5. ExcelReader读取所以sheet页。

2.2.6 测试多sheet页，多head

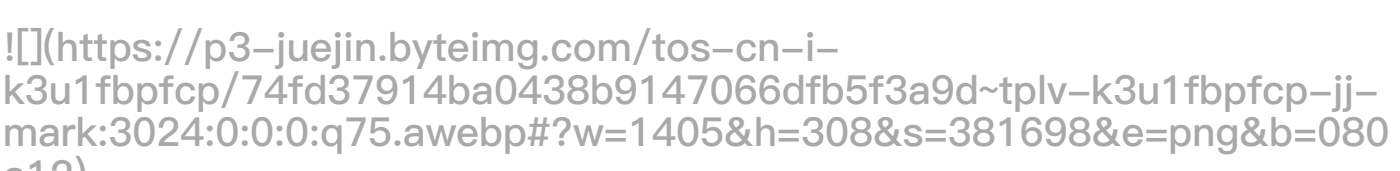
上述工具基本实现了多sheet页，多head读取数据，那么我们来测试一下。

```
/**
 * @author: jiangjs
 * @description:
 * @date: 2023/12/26 15:38
 **/
@Service
@Slf4j
public class UserServiceImpl extends ServiceImpl<UserMapper, User> implements UserService{

    private final static List<String> EXCEL_SUFFIX = Arrays.asList(”xls”,”xlsx”);

    @Override
    public void importDataByExcel(MultipartFile file) {
        String filename = file.getOriginalFilename();
        Assert.notNull(filename,”文件名称不能为空”);
        String suffix = filename.substring(filename.lastIndexOf(”.”)+1);
        if (!EXCEL_SUFFIX.contains(suffix)){
            throw new RuntimeException(”上传的文件不是Excel文件!”);
```

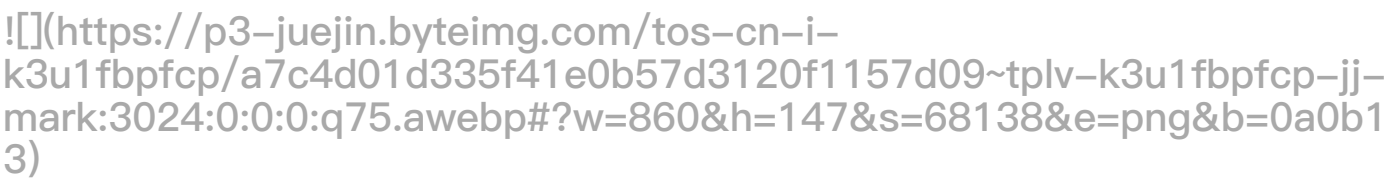
```
        }
        ExcelImportParamVO importParamVO = new
ExcelImportParamVO();
importParamVO.setFile(file).setClazz(User.class).setReadListener(new Pu
blicReadExcelListener<>(this))
                .setSheetNum(2)
                .setHeadNums(new Integer[]{1,2});
        EasyExcelUtil.importExcelFile(importParamVO);
    }
}

```

准备表格，sheet1设置head从1开始，内容：赵六，sheet2设置head从2开始，内容：周七，如图：

执行测试结果：



数据库结果：



执行结果实现了多sheet页，指定多行head。

上述就是我在使用EasyExcel做的一些封装，基本能做到通用的导入，小伙伴们觉得有用的话，记得点赞、收藏哦，如果在使用过程中大家还有啥其他想法，评论区大家可以说说。

谢谢大家听我唠叨，我是抢老婆酸奶的小肥仔，下次见.....

原文链接: https://juejin.cn/post/7350199693753663539