EasyExcel，让excel导入导出更加简单

EasyExcel

> 在做excel导入导出的时候,发现项目中封装的工具类及其难用,于是去gitHub上找了一些相关的框架,最终选定了EasyExcel。之前早有听闻该框架，但是一直没有去了解，这次借此学习一波，提高以后的工作效率。实际使用中，发现是真的很easy，大部分api通过名称就能知道大致意思，这点做的很nice。参考文档，大部分场景的需求基本都能够满足。

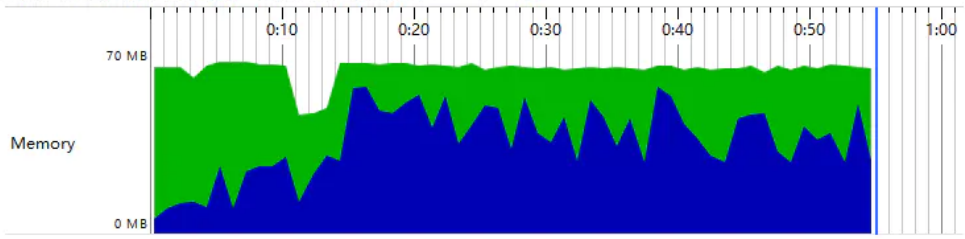
GitHub上的官方说明

JAVA解析Excel工具easyexcel

Java解析、生成Excel比较有名的框架有Apache poi、jxl。但他们都存在一个严重的问题就是非常的耗内存，poi有一套SAX模式的API可以一定程度的解决一些内存溢出的问题，但POI还是有一些缺陷，比如07版Excel解压缩以及解压后存储都是在内存中完成的，内存消耗依然很大。easyexcel重写了poi对07版Excel的解析，能够原本一个3M的excel用POI sax依然需要100M左右内存降低到几M，并且再大的excel不会出现内存溢出，03版依赖POI的sax模式。在上层做了模型转换的封装，让使用者更加简单方便

64M内存1分钟内读取75M(46W行25列)的Excel

当然还有急速模式能更快，但是内存占用会在100M多一点



image

快速开始

maven仓库地址

```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>easyexcel</artifactId>
    <version>2.1.2</version>
</dependency>
```

导入

如下图excel表格：

	A	B	C	D	E	F
1	额度利率导入模板表					
2	客户名称	MIS编码	年度最高额（元）	月度滚动额（元）	最新应收账款余额（元）	本月利率(年化）
3	安徽省盛格新能源科技有限公司	1111	11,111,100.00	11,111,100.00	11,111,100.00	9.50%
4	安徽省盛格新能源科技有限公司2	2222	11,111,100.00	11,111,100.00	11,111,100.00	9.50%
5						
6	安徽省盛格新能源科技有限公司3	333	11,111,100.00		11,111,100.00	9.50%
7						
8						

image

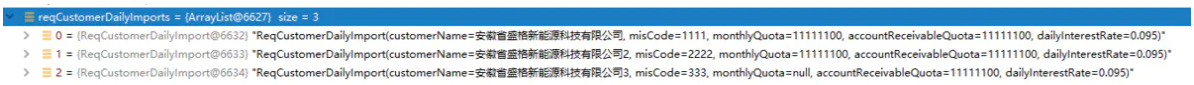
建立导入对应实体类

```java
@Data
public class ReqCustomerDailyImport {
    /**
     * 客户名称
     */
    @ExcelProperty(index = 0)
    private String customerName;

    /**
     * MIS编码
     */
    @ExcelProperty(index = 1)
    private String misCode;

    /**
     * 月度滚动额
     */
    @ExcelProperty(index = 3)
    private BigDecimal monthlyQuota;

    /**
     * 最新应收账款余额
     */
    @ExcelProperty(index = 4)
    private BigDecimal accountReceivableQuota;

    /**
     * 本月利率(年化）
     */
    @ExcelProperty(index = 5)
    private BigDecimal dailyInterestRate;
}
```

Controller代码

```
@PostMapping("/import")
public void importCustomerDaily(@RequestParam MultipartFile file) throws IOException {
    InputStream inputStream = file.getInputStream();
    List<ReqCustomerDailyImport> reqCustomerDailyImports = EasyExcel.read(inputStream)
            .head(ReqCustomerDailyImport.class)
            // 设置sheet,默认读取第一个
            .sheet()
            // 设置标题所在行数
            .headRowNumber(2)
            .doReadSync();
}
```

运行结果



image

> 可以看出只需要在实体对象使用@ExcelProperty注解，读取时指定该class，即可读取，并且自动过滤了空行，对于excel的读取及其简单。不过此时发现一个问题，这样我如果要校验字段该怎么办？要将字段类型转换成另外一个类型呢？
> 不必担心，我们可以想到的问题，作者肯定也考虑到了，下面来一个Demo

代码如下

```
List<ReqCustomerDailyImport> reqCustomerDailyImports = EasyExcel.read(inputStream)
            // 这个转换是成全局的， 所有java为string,excel为string的都会用这个转换器。
            // 如果就想单个字段使用请使用@ExcelProperty 指定converter
            .registerConverter(new StringConverter())
            // 注册监听器，可以在这里校验字段
            .registerReadListener(new CustomerDailyImportListener())
            .head(ReqCustomerDailyImport.class)
            .sheet()
            .headRowNumber(2)
            .doReadSync();
}
```

监听器

```
public class CustomerDailyImportListener extends AnalysisEventListener {

    List misCodes = Lists.newArrayList();

    /**
     * 每解析一行，回调该方法
     * @param data
```

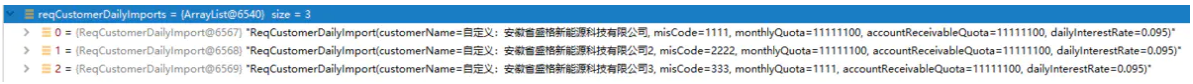
```
     * @param context
     */
    @Override
    public void invoke(Object data, AnalysisContext context) {
        String misCode = ((ReqCustomerDailyImport) data).getMisCode();
        if (StringUtils.isEmpty(misCode)) {
            throw new RuntimeException(String.format("第%s行MIS编码为空，请核实", context.readRowHolder().getRowIndex() + 1));
        }
        if (misCodes.contains(misCodes)) {
            throw new RuntimeException(String.format("第%s行MIS编码已重复，请核实", context.readRowHolder().getRowIndex() + 1));
        } else {
            misCodes.add(misCode);
        }
    }

    /**
     * 出现异常回调
     * @param exception
     * @param context
     * @throws Exception
     */
    @Override
    public void onException(Exception exception, AnalysisContext context) throws Exception {
        // ExcelDataConvertException:当数据转换异常的时候，会抛出该异常，此处可以得知第几行，第几列的数据
        if (exception instanceof ExcelDataConvertException) {
            Integer columnIndex = ((ExcelDataConvertException) exception).getColumnIndex() + 1;
            Integer rowIndex = ((ExcelDataConvertException) exception).getRowIndex() + 1;
            String message = "第" + rowIndex + "行，第" + columnIndex + "列" + "数据格式有误，请核实";
            throw new RuntimeException(message);
        } else if (exception instanceof RuntimeException) {
            throw exception;
        } else {
            super.onException(exception, context);
        }
    }

    /**
     * 解析完全部回调
     * @param context
     */
    @Override
    public void doAfterAllAnalysed(AnalysisContext context) {
        misCodes.clear();
    }
}
```

转换器

```java
public class StringConverter implements Converter<String> {

    @Override
    public Class supportJavaTypeKey() {
        return String.class;
    }

    @Override
    public CellDataTypeEnum supportExcelTypeKey() {
        return CellDataTypeEnum.STRING;
    }

    /**
     * 将excel对象转成Java对象，这里读的时候会调用
     *
     * @param cellData            NotNull
     * @param contentProperty     Nullable
     * @param globalConfiguration NotNull
     * @return
     */
    @Override
    public String convertToJavaData(CellData cellData, ExcelContentProperty contentProperty,
                                    GlobalConfiguration globalConfiguration) {
        return "自定义：" + cellData.getStringValue();
    }

    /**
     * 将Java对象转成String对象，写出的时候调用
     *
     * @param value
     * @param contentProperty
     * @param globalConfiguration
     * @return
     */
    @Override
    public CellData convertToExcelData(String value, ExcelContentProperty contentProperty,
                                       GlobalConfiguration globalConfiguration) {
        return new CellData(value);
    }
}
```

> 可以看出注册了一个监听器：CustomerDailyImportListener,还注册了一个转换器：StringConverter。流程为：框架读取一行数据，先执行转换器，当一行数据转换完成，执行监听器的回调方法，如果转换的过程中，出现转换异常，也会回调监听器中的onException方法。因此，可以在监听器中校验数据，在转换器中转换数据类型或者格式。

运行结果



image

修改一下表格，测试校验是否生效

	额度利率导入模板表					
2	客户名称	MIS编码	年度最高额（元）	月度滚动额（元）	最新应收账款余额（元）	本月利率(年化）
3	安徽省盛格新能源科技有限公司		11,111,100.00	11,111,100.00	11,111,100.00	9.50%
4	安徽省盛格新能源科技有限公司2	2222	11,111,100.00	11,111,100.00	11,111,100.00	9.50%
5						
6	安徽省盛格新能源科技有限公司3	333	11,111,100.00	1,111.00	11,111,100.00	9.50%

image

再次导入，查看运行结果

image

导入相关常用API

注解

- `ExcelProperty` 指定当前字段对应excel中的那一列。可以根据名字或者Index去匹配。当然也可以不写，默认第一个字段就是index=0，以此类推。千万注意，要么全部不写，要么全部用index，要么全部用名字去匹配。千万别三个混着用，除非你非常了解源代码中三个混着用怎么去排序的。
- `ExcelIgnore` 默认所有字段都会和excel去匹配，加了这个注解会忽略该字段。
- `DateTimeFormat` 日期转换，用String去接收excel日期格式的数据会调用这个注解。里面的value参照java.text.SimpleDateFormat。
- `NumberFormat` 数字转换，用String去接收excel数字格式的数据会调用这个注解。里面的value参照java.text.DecimalFormat。

EasyExcel相关参数

- `readListener` 监听器，在读取数据的过程中会不断的调用监听器。
- `converter` 转换器，默认加载了很多转换器。也可以自定义，如果使用的是registerConverter，那么该转换器是全局的，如果要对单个字段生效，可以在ExcelProperty注解的converter指定转换器。
- `headRowNumber` 需要读的表格有几行头数据。默认有一行头，也就是认为第二行开始起为数据。
- head 与clazz二选一。读取文件头对应的列表，会根据列表匹配数据，建议使用class。
- `autoTrim` 字符串、表头等数据自动trim。
- `sheetNo` 需要读取Sheet的编码，建议使用这个来指定读取哪个Sheet。
- `sheetName` 根据名字去匹配Sheet,excel 2003不支持根据名字去匹配。

导出

建立导出对应实体类

```java
@Data
@Builder
public class RespCustomerDailyImport {

    @ExcelProperty("客户编码")
    private String customerName;

    @ExcelProperty("MIS编码")
    private String misCode;

    @ExcelProperty("月度滚动额")
    private BigDecimal monthlyQuota;

    @ExcelProperty("最新应收账款余额")
    private BigDecimal accountReceivableQuota;

    @NumberFormat("#.##%")
    @ExcelProperty("本月利率(年化）")
    private BigDecimal dailyInterestRate;
}
```

Controller代码

```java
@GetMapping("/export")
public void export(HttpServletResponse response) throws IOException {
    // 生成数据
    List<RespCustomerDailyImport> respCustomerDailyImports =
Lists.newArrayList();
    for (int i = 0; i < 50; i++) {
        RespCustomerDailyImport respCustomerDailyImport =
RespCustomerDailyImport.builder()
                .misCode(String.valueOf(i))
                .customerName("customerName" + i)
                .monthlyQuota(new BigDecimal(String.valueOf(i)))
                .accountReceivableQuota(new BigDecimal(String.valueOf(i)))
                .dailyInterestRate(new
BigDecimal(String.valueOf(i))).build();
        respCustomerDailyImports.add(respCustomerDailyImport);
    }

    response.setContentType("application/vnd.ms-excel");
    response.setCharacterEncoding("utf-8");
    // 这里URLEncoder.encode可以防止中文乱码 当然和easyexcel没有关系
    String fileName = URLEncoder.encode("导出", "UTF-8");
    response.setHeader("Content-disposition", "attachment;filename=" +
fileName + ".xlsx");
    EasyExcel.write(response.getOutputStream(),
RespCustomerDailyImport.class)
            .sheet("sheet0")
            // 设置字段宽度为自动调整，不太精确
```

```
            .registerWriteHandler(new
LongestMatchColumnWidthStyleStrategy())
            .doWrite(respCustomerDailyImports);
}
```

导出效果

	A	B	C	D	E
1	客户编码	MIS编码	月度滚动额	最新应收账款余额	本月利率(年化)
2	customerName0	0	0	0	0%
3	customerName1	1	1	1	100%
4	customerName2	2	2	2	200%
5	customerName3	3	3	3	300%
6	customerName4	4	4	4	400%
7	customerName5	5	5	5	500%
8	customerName6	6	6	6	600%
9	customerName7	7	7	7	700%
10	customerName8	8	8	8	800%
11	customerName9	9	9	9	900%
12	customerName10	10	10	10	1000%
13	customerName11	11	11	11	1100%
14	customerName12	12	12	12	1200%
15	customerName13	13	13	13	1300%
16	customerName14	14	14	14	1400%
17	customerName15	15	15	15	1500%
18	customerName16	16	16	16	1600%
19	customerName17	17	17	17	1700%
20	customerName18	18	18	18	1800%
21	customerName19	19	19	19	1900%
22	customerName20	20	20	20	2000%
23	customerName21	21	21	21	2100%
24	customerName22	22	22	22	2200%
25	customerName23	23	23	23	2300%
26	customerName24	24	24	24	2400%
27	customerName25	25	25	25	2500%
28	customerName26	26	26	26	2600%
29	customerName27	27	27	27	2700%
30	customerName28	28	28	28	2800%
31	customerName29	29	29	29	2900%
32	customerName30	30	30	30	3000%
33	customerName31	31	31	31	3100%
34	customerName32	32	32	32	3200%
35	customerName33	33	33	33	3300%
36	customerName34	34	34	34	3400%
37	customerName35	35	35	35	3500%
38	customerName36	36	36	36	3600%
39	customerName37	37	37	37	3700%
40	customerName38	38	38	38	3800%
41	customerName39	39	39	39	3900%
42	customerName40	40	40	40	4000%

sheet0

image

导出相关常用API

注解

- `ExcelProperty` 指定写到第几列，默认根据成员变量排序。value指定写入的名称，默认成员变量的名字。
- `ExcelIgnore` 默认所有字段都会写入excel，这个注解会忽略这个字段。

- `DateTimeFormat` 日期转换，将Date写到excel会调用这个注解。里面的value参照java.text.SimpleDateFormat。
- `NumberFormat` 数字转换，用Number写excel会调用这个注解。里面的value参照java.text.DecimalFormat。

EasyExcel相关参数

- `needHead` 监听器是否导出头。
- `useDefaultStyle` 写的时候是否是使用默认头。
- head 与clazz二选一。写入文件的头列表，建议使用class。
- `autoTrim` 字符串、表头等数据自动trim。
- `sheetNo` 需要写入的编码。默认0。
- `sheetName` 需要些的Sheet名称，默认同sheetNo。

总结

> 可以看出不管是excel的读取还是写入，都是一个注解加上一行代码完成，可以让我们少些很多解析的代码，极大减少了重复的工作量。当然这两个例子使用了最简单的方式，EasyExcel还支持更多场景，例如读，可以读多个sheet，也可以解析一行数据或者多行数据做一次入库操作；写的话，支持复杂头，指定列写入，重复多次写入，多个sheet写入，根据模板写入等等。更多的例子可以去参考EasyExcel官方文档