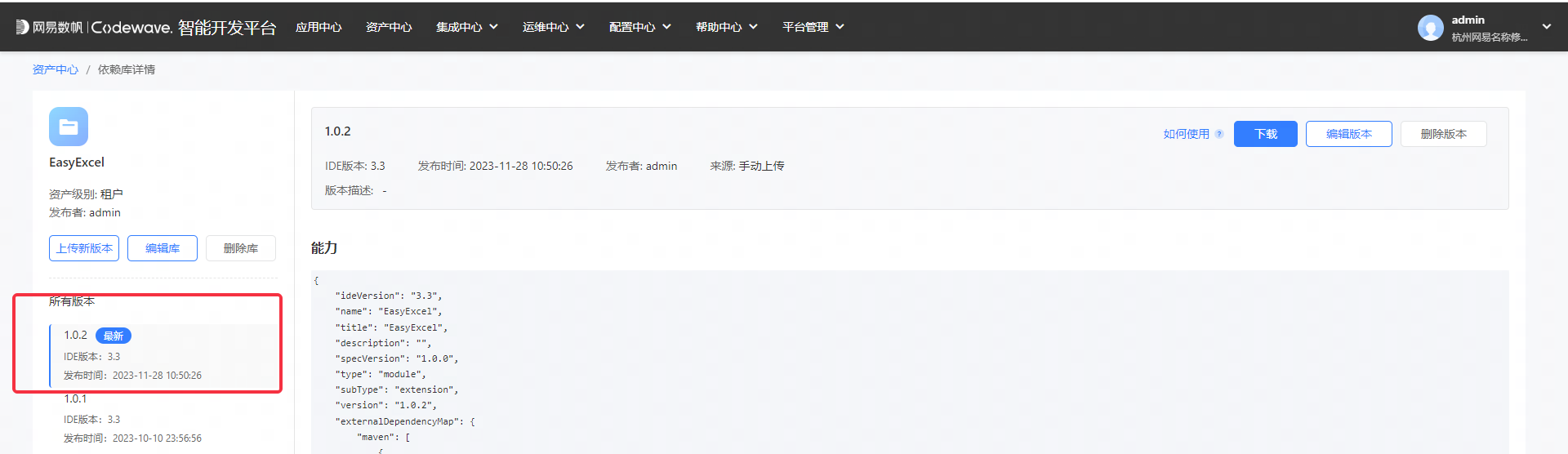
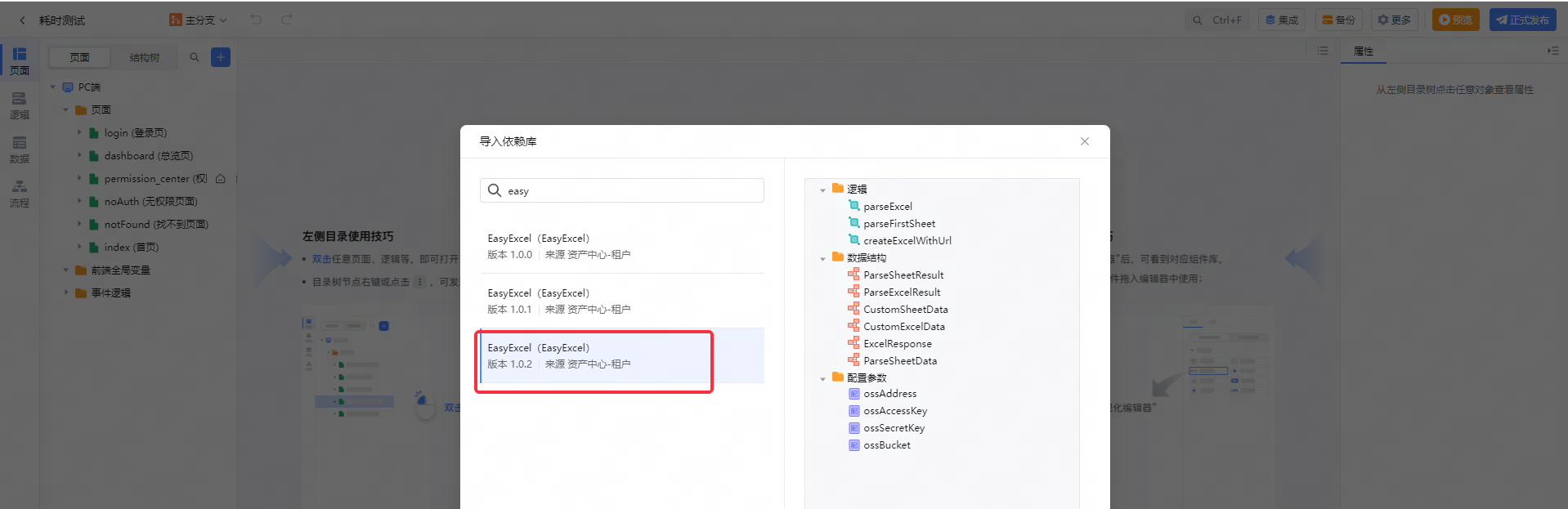
**1. 获取最新版依赖库**

注意检查当前租户下依赖库最新版本，防止出现版本号冲突



**2. 导入依赖库**



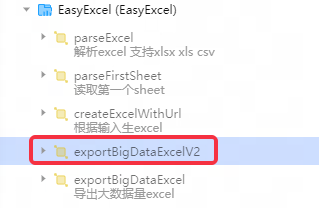
**3. 针对大数据量导入导出**

**3.1. 数据导出到excel-v2**

v1版本的数据导出操作上较为繁琐，如果导出报表字段较多，可以使用v2逻辑。

|  |  |
| --- | --- |
| 逻辑 | 说明 |
| exportBigDataExcel | 参见导出实践-v1 |
| exportBigDataExcelV2 | 简化了表头、数据的插入 |

依赖库版本需要大于2.1.9，导入后可以看到v2逻辑。



**3.1.1 拖入exportBigDataExcelV2**



参数说明：

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 说明 |
| queryData | Function<QueryCondition,List<String>> | 用户自定义数据查询逻辑 |
| condition | QueryCondition | 数据查询条件 |
| saveResult | Function<ExportBigDataResponse,Boolean>> | 用户自定义回调通知逻辑 |

返回值：

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 说明 |
| result | ExportBigDataResponse | 同步返回信息  当设置为同步模式，返回excel导出结果；  当设置为异步模式，返回任务受理信息。 |

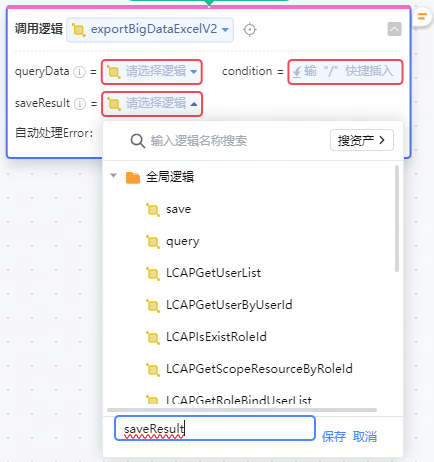
**3.1.2 添加queryData逻辑**



点击queryData参数值，在下拉框中添加queryData逻辑，保存即可。（queryData这个名字可依据业务自定义）

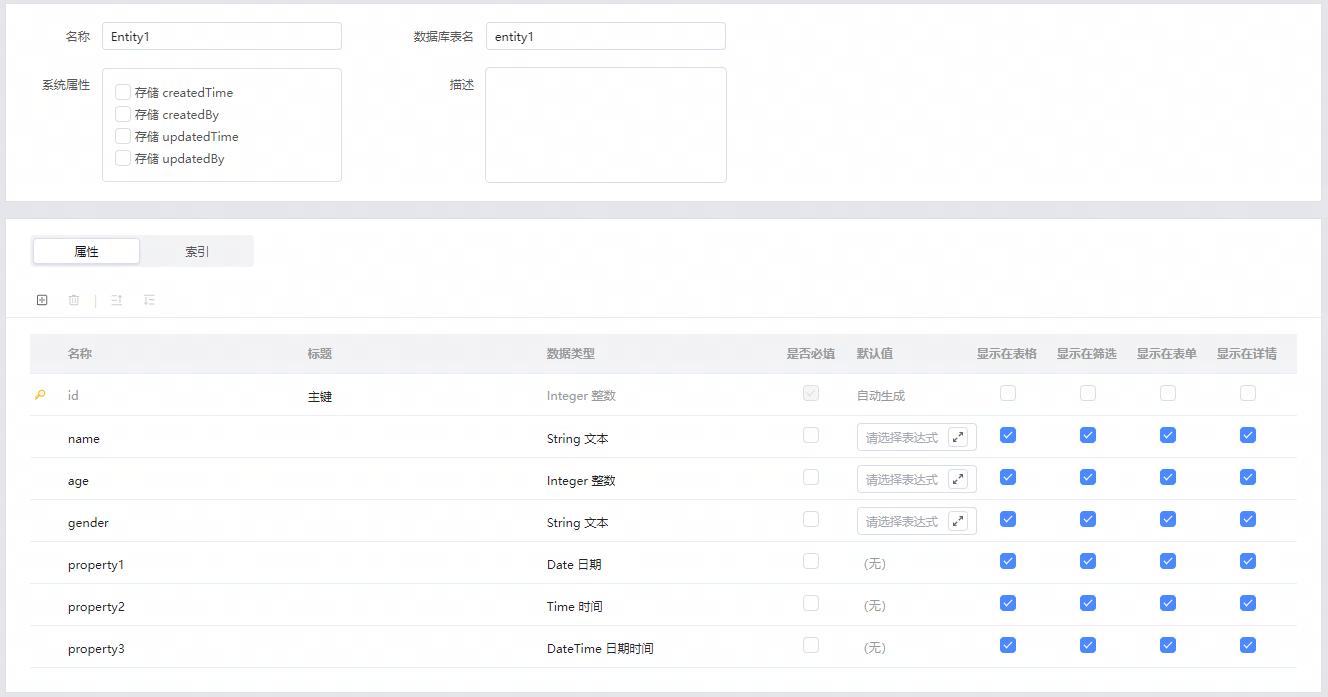
**3.1.3 添加saveResult逻辑**

操作同上，点击saveResult参数值，在下拉框中添加saveResult逻辑，保存即可。（如果使用同步，不需要实现saveResult逻辑内容，留空即可）



**3.1.4 定义导出数据结构**

本例中，我们将实体Entity1中数据导出到excel，实体属性如下：

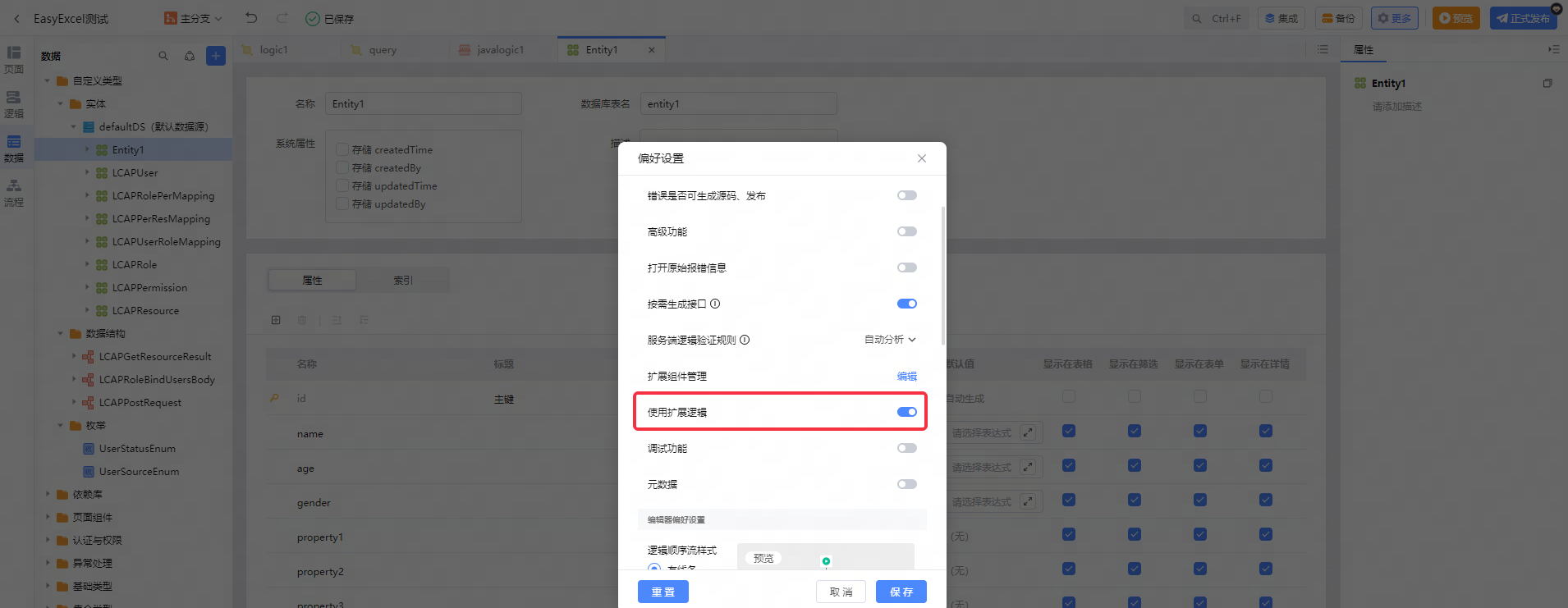


**注意属性名称与数据类型**！**导出数据结构需要保持完全一致**!

下面通过“扩展逻辑”定义导出数据结构：

**3.1.4.1 开启扩展逻辑**

在“系统偏好设置”打开“扩展逻辑"

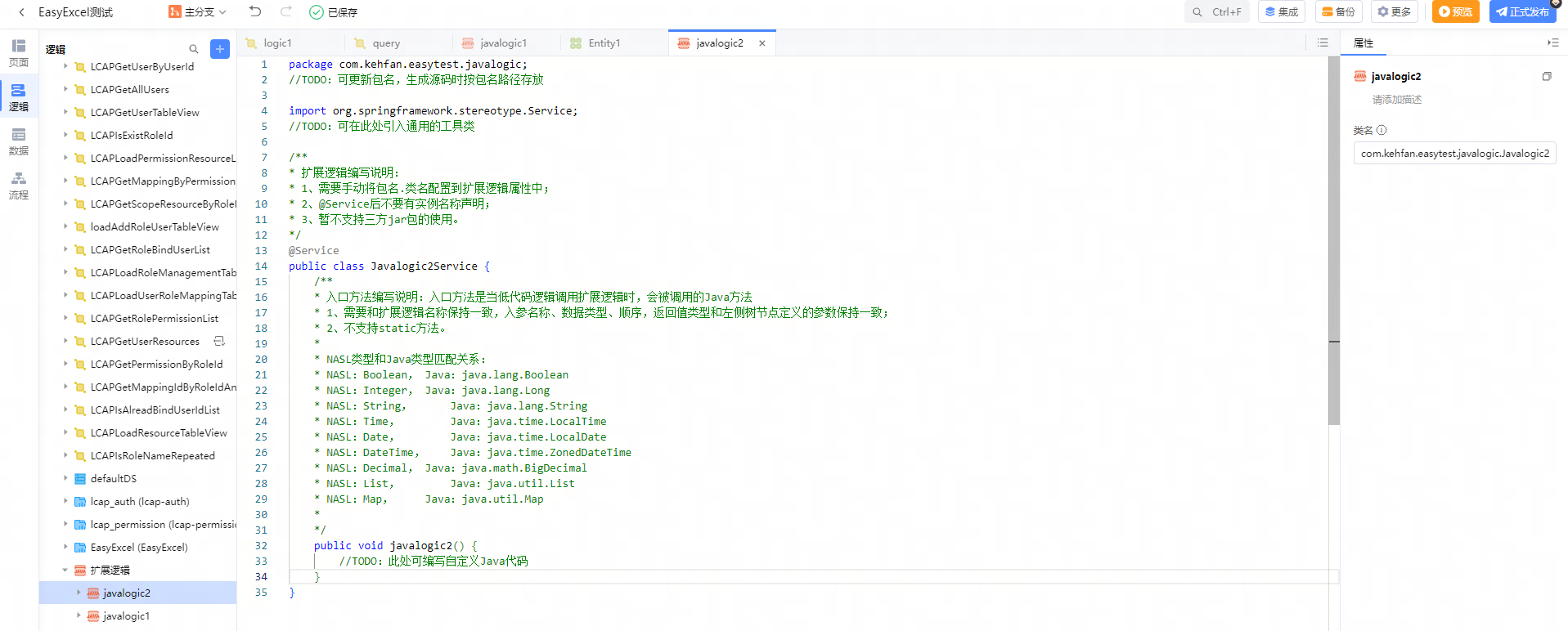


开启后，在服务端逻辑列表末尾，可以看到：

https://office.netease.com/api/admin/file/download?path=cowork/2024/03/26/5bf4a8f2dafb43aaa49e8abe8c3eed4b.png

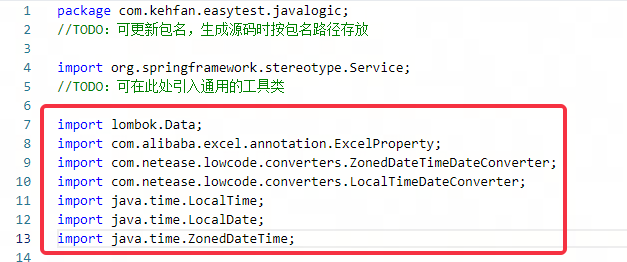
**3.1.4.2 定义导出数据结构**

添加扩展逻辑，会自动生成模板代码：



我们仅需添加以下几个部分：

添加import



可直接复制如下代码：

import lombok.Data;

import com.alibaba.excel.annotation.ExcelProperty;

import com.netease.lowcode.converters.ZonedDateTimeDateConverter;

import com.netease.lowcode.converters.LocalTimeDateConverter;

import java.time.LocalTime;

import java.time.LocalDate;

import java.time.ZonedDateTime;

添加数据结构



可以直接复制如下代码：

    @Data

    public static class ModelData {

        @ExcelProperty({"主标题", "姓名1"})

        String name;

        @ExcelProperty({"主标题", "年龄2"})

        Integer age;

        @ExcelProperty({"主标题", "性别3"})

        String gender;

        @ExcelProperty({"主标题2", "date类型"})

        LocalDate property1;

        @ExcelProperty(value = {"主标题2", "time类型"},converter = LocalTimeDateConverter.class)

        LocalTime property2;

        @ExcelProperty(value = {"主标题2", "datetime类型"},converter = ZonedDateTimeDateConverter.class)

        ZonedDateTime property3;

    }

注意数据结构添加到扩展逻辑类内部，数据结构类上添加`@Data` 注解。

属性名、数据类型与结构体保持完全一致，对于几个特殊类型，请按照如下映射选择：

    \* NASL类型和Java类型匹配关系：

    \* NASL：Boolean， Java：java.lang.Boolean

    \* NASL：Integer， Java：java.lang.Long

    \* NASL：String，      Java：java.lang.String

    \* NASL：Time，        Java：java.time.LocalTime

    \* NASL：Date，        Java：java.time.LocalDate

    \* NASL：DateTime，    Java：java.time.ZonedDateTime

    \* NASL：Decimal， Java：java.math.BigDecimal

    \* NASL：List，        Java：java.util.List

    \* NASL：Map，     Java：java.util.Map

比如：我的结构体属性property3的数据类型为DateTime，那么我在扩展逻辑中定义就要定义成ZonedDateTime

关于表头设置，在属性上添加注解`@ExcelProperty`，支持多行表头，并且支持合并表头：

单行表头示例:

@ExcelProperty({"表头1"})

@ExcelProperty({"表头2"})

@ExcelProperty({"表头3"})

多行合并表头示例:

@ExcelProperty({"主标题", "姓名1"})

@ExcelProperty({"主标题", "年龄2"})

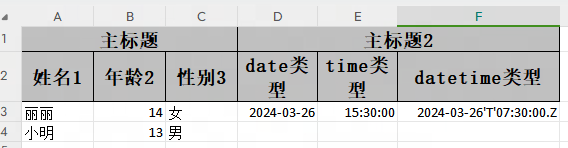
@ExcelProperty({"主标题", "性别3"})

@ExcelProperty({"主标题2", "date类型"})

第一列对应excel的第一行表头，如果名称相同则会自动合并；

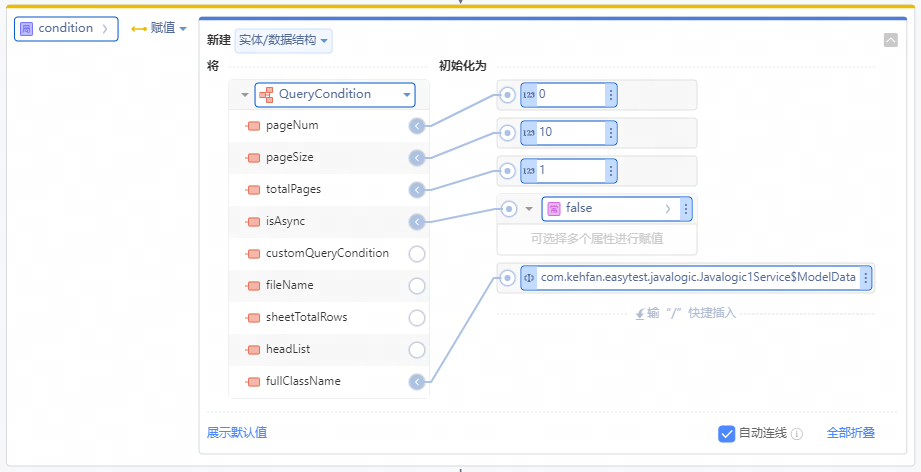
第二列对应excel的第二行表头；

表头效果:



**3.1.5. 设置Condition**

仅需设置必要参数，结构为：

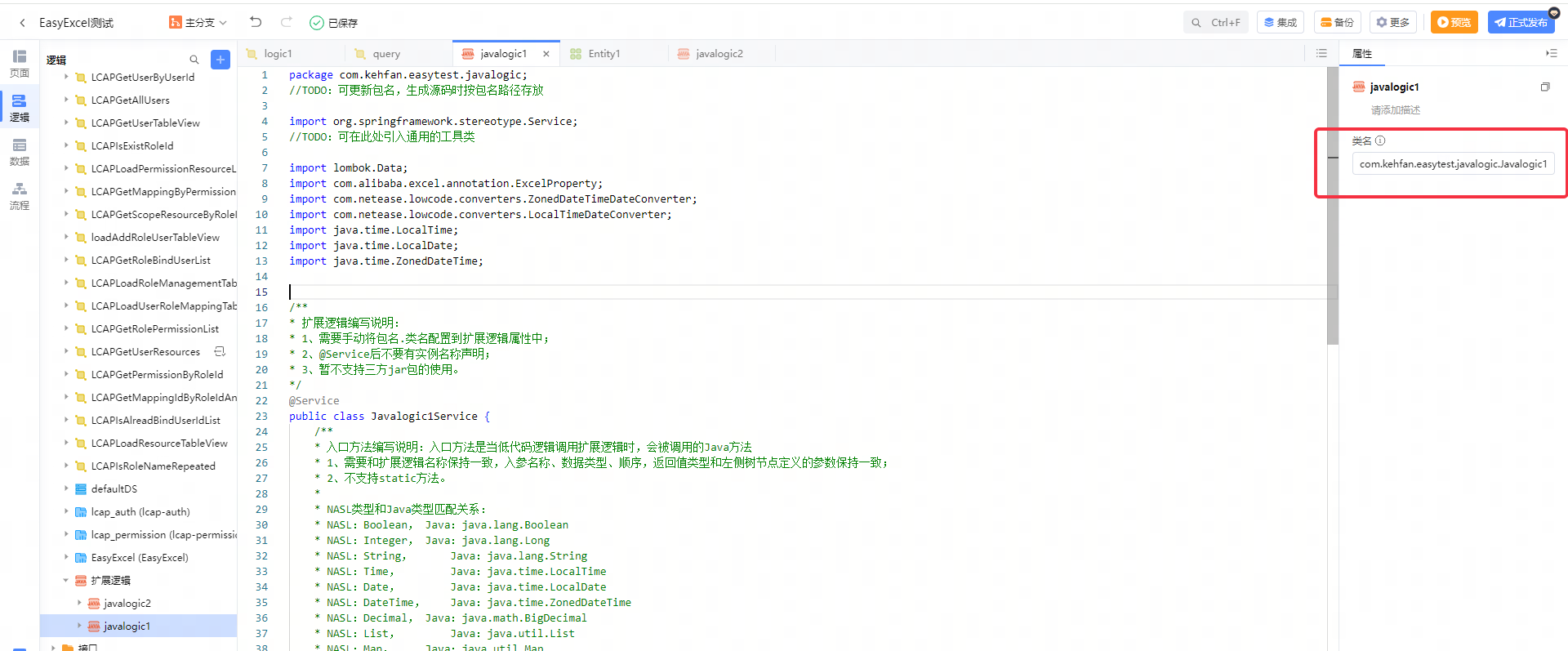


参数说明:

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| pageNum | integer | 从0开始，分页信息 |
| pageSize | integer | 每页数据大小 |
| totalPages | integer | 总页数 |
| isAsync | boolean | 是否异步：  false同步，直接返回结果；  true异步，需要在saveResult处理结果 |
| customQueryCondition | Map<String,String> | 查询参数，如有额外查询参数，请通过这个传递给query逻辑 |
| fileName | String | excel文件名 |
| sheetTotalRows | integer | 每个sheet填充多少数据 |
| **fullClassName** | **string** | **导出数据结构的全类名** |

**3.1.5.1 fullClassName设置**

在扩展逻辑页面右侧，复制类名：com.kehfan.easytest.javalogic.Javalogic1Service



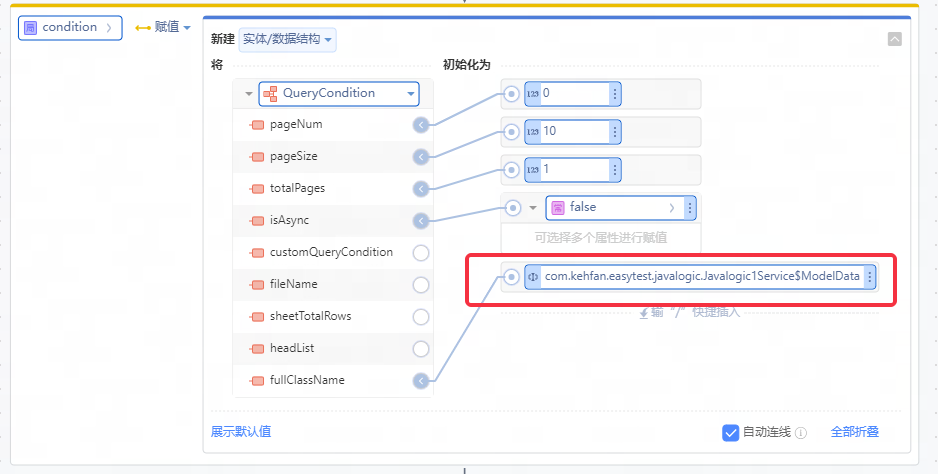
我们定义的导出结构类名为：ModelData



将类名与结构类名通过$连接得到：

**com.kehfan.easytest.javalogic.Javalogic1Service$ModelData**

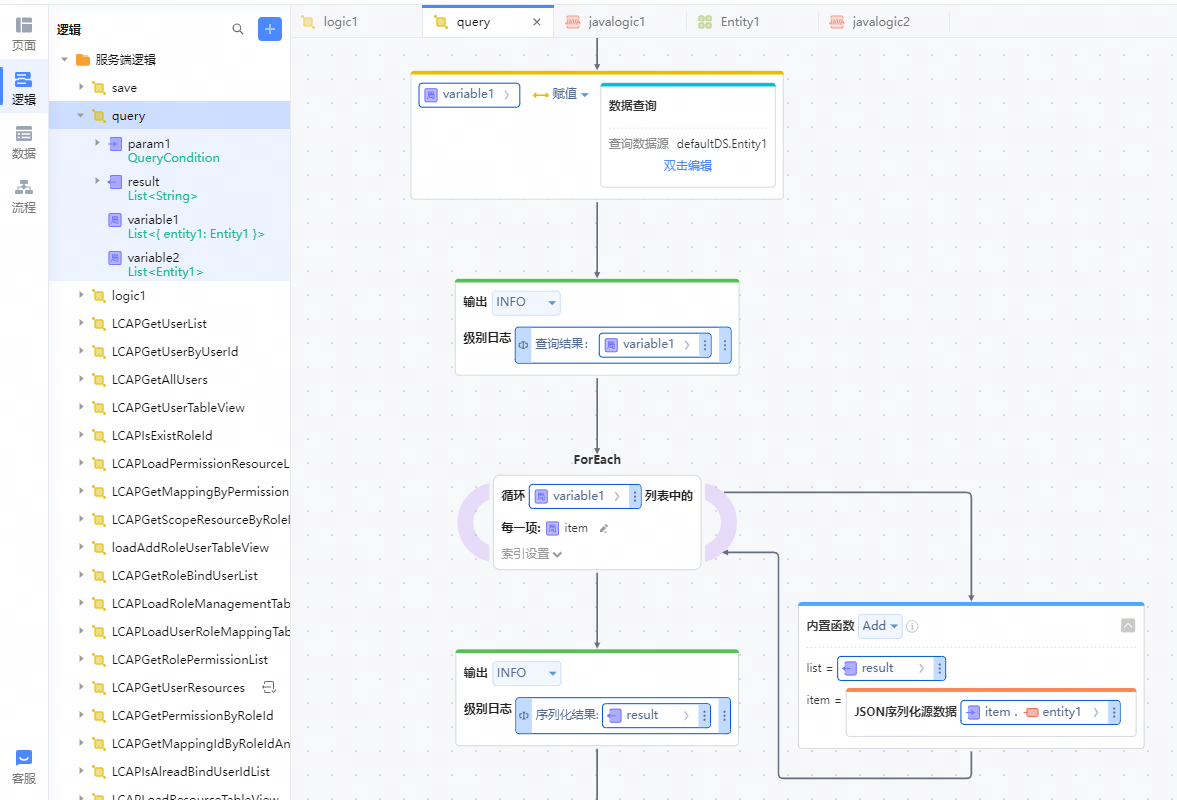
将这个字符串赋值到condition的fullClassName



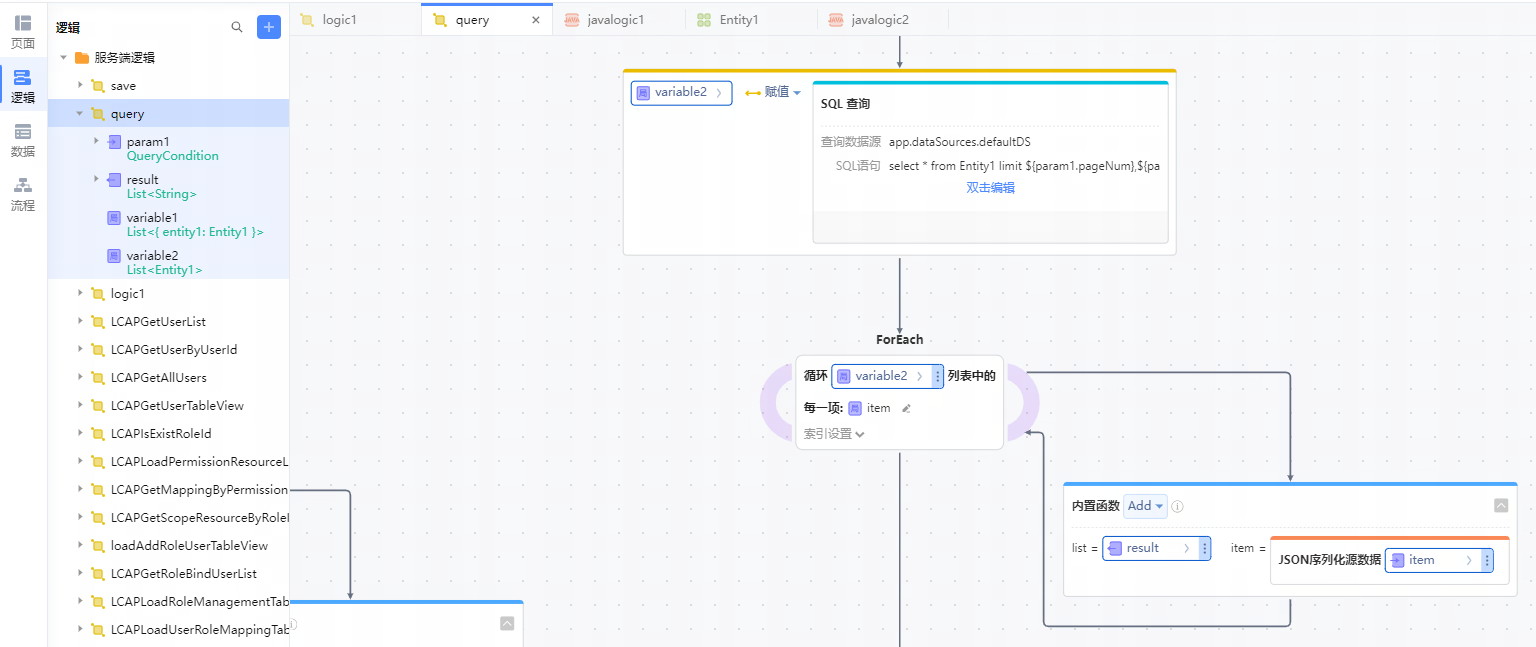
**3.1.6. 实现查询逻辑**

自动生成的query逻辑中添加数据查询逻辑，本例以数据库查询作为参考：

方式1



方式2



无论哪种方式，都需要将结果序列化后添加到result中。

注意：**第一种方式中需要使用item.entity1拿到真正的结构体后再序列化**。

**3.1.7. 日期类型**

注意：导出数据结构内包含Time(Nasl,Java类型为java.time.LocalTime)、DateTime(Nasl,Java类型为java.time.ZonedDateTime)、汇报错如下：



原因：底层框架不支持这两种类型，需要手动指定converter！

依赖库内已经实现了上述两种类型转换，分别为：

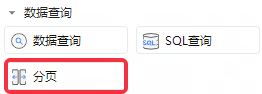
com.netease.lowcode.converters.LocalTimeDateConverter

com.netease.lowcode.converters.ZonedDateTimeDateConverter

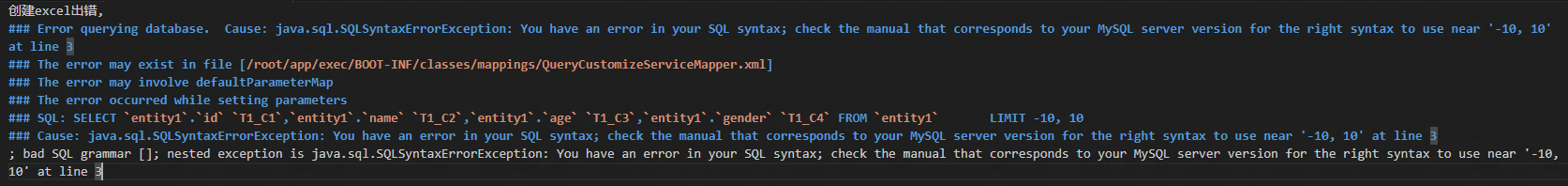
使用方式请参考:



**3.1.8. 不支持分页组件**



在query逻辑中不要使用“分页”组件，导出会报错

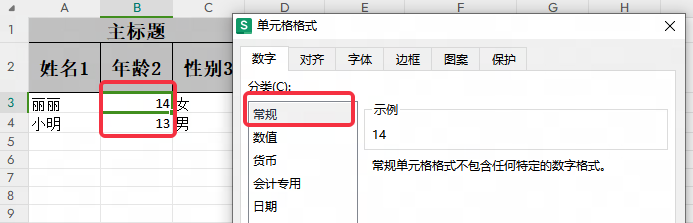


如需分页，请使用“SQL查询”组件，通过limit实现分页。

**3.1.9 设置单元格格式**

**3.1.9.1 设置数值类型**

默认导出的excel，单元格类型为常规。



我们如有特殊需求，希望某个列为数值列，且保留指定小数位：



如上图，

导入注解

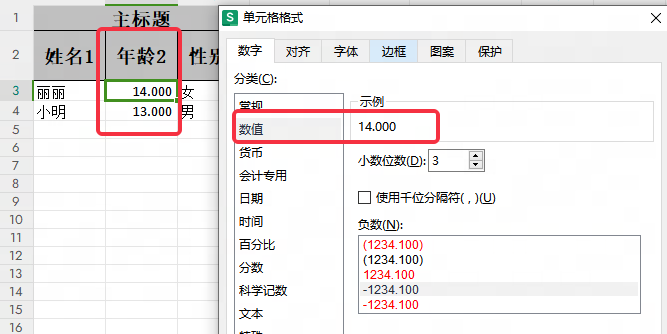
import com.alibaba.excel.annotation.format.NumberFormat;

增加注解@NumberFormat("0.000\_ ")

0.000标识保留三位小数，

\_ 表示会更改单元格类型为数值，注意：\_后面有一个空格！

效果如下：



**3.2 excel导入解析**

library-EasyExcel-2.1.12 开始支持。

逻辑：parseBigDataExcel

入参：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| handle | Function<List<String>,String> | 用户自定义数据处理逻辑 |
| request | ParseRequest | 请求信息 |

结构体：ParseRequest

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| url | string | 上传的excel地址 |
| fullClassName | string | 需要解析的结构类名称 |

返回结构体：ParseBigDataResponse

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| success |  |  |
| msg |  |  |

**操作指南**

由于包含回调，该逻辑需要在后端逻辑中使用。

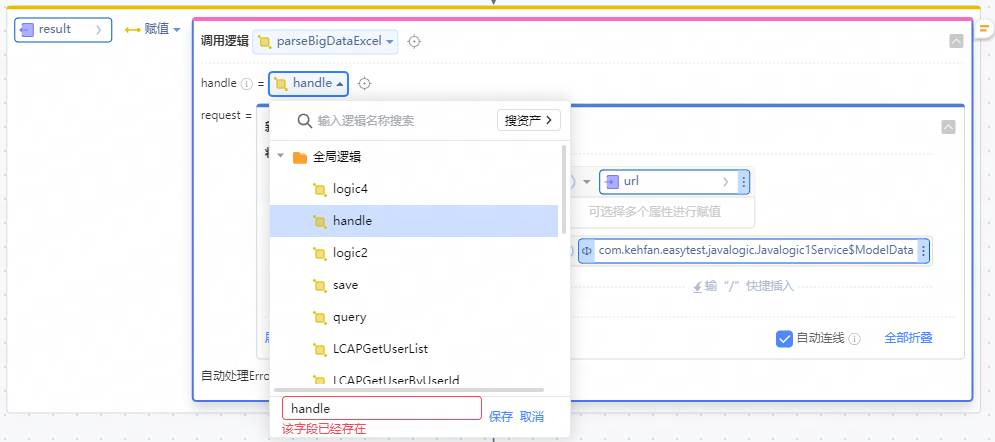
1. 拖入parseBigDataExcel逻辑。



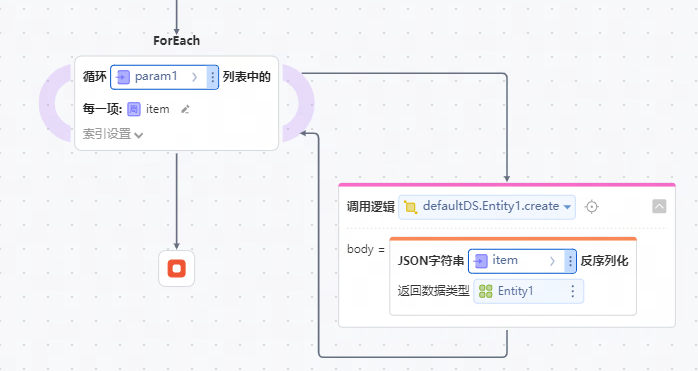
2. 设置request参数。



3. 新增handle逻辑处理数据



4. 遍历param1列表，反序列化后写入数据库。



5. 关于fullClassName，参考数据导出v2，定义扩展逻辑。

**定义扩展逻辑**

1. 首先在系统设置里开启扩展逻辑。

2. 添加一个扩展逻辑。

3. 导入所需依赖

import lombok.Data;

// 如果excel有图片列，需要在图片对应的属性上加此注解

import com.netease.lowcode.annotation.Picture;

import com.alibaba.excel.annotation.ExcelProperty;

4. 代码示例

package com.kehfan.parseimage.javalogic;

import org.springframework.stereotype.Service;

//TODO：可在此处引入通用的工具类

import lombok.Data;

import com.netease.lowcode.annotation.Picture;

import com.alibaba.excel.annotation.ExcelProperty;

@Service

public class Javalogic1Service {

// 默认生成的，保持原样即可

    public void javalogic1() {

        //TODO：此处可编写自定义Java代码

    }

// 在默认逻辑下方添加excel解析的数据结构

    @Data

    public static class TestDataModel {

// 通过ExcelProperty注解绑定excel的表头与属性名称

        @ExcelProperty({"姓名"})

        private String name;

        @ExcelProperty({"年龄"})

        private Integer age;

// 添加注解表示这个列是一个图片列，并指出列的索引（从0开始）

        @Picture(columnIndex = 2)

        private String imageUrl;

    }

}

**3.3. 数据导出到excel-v1**

**概述**

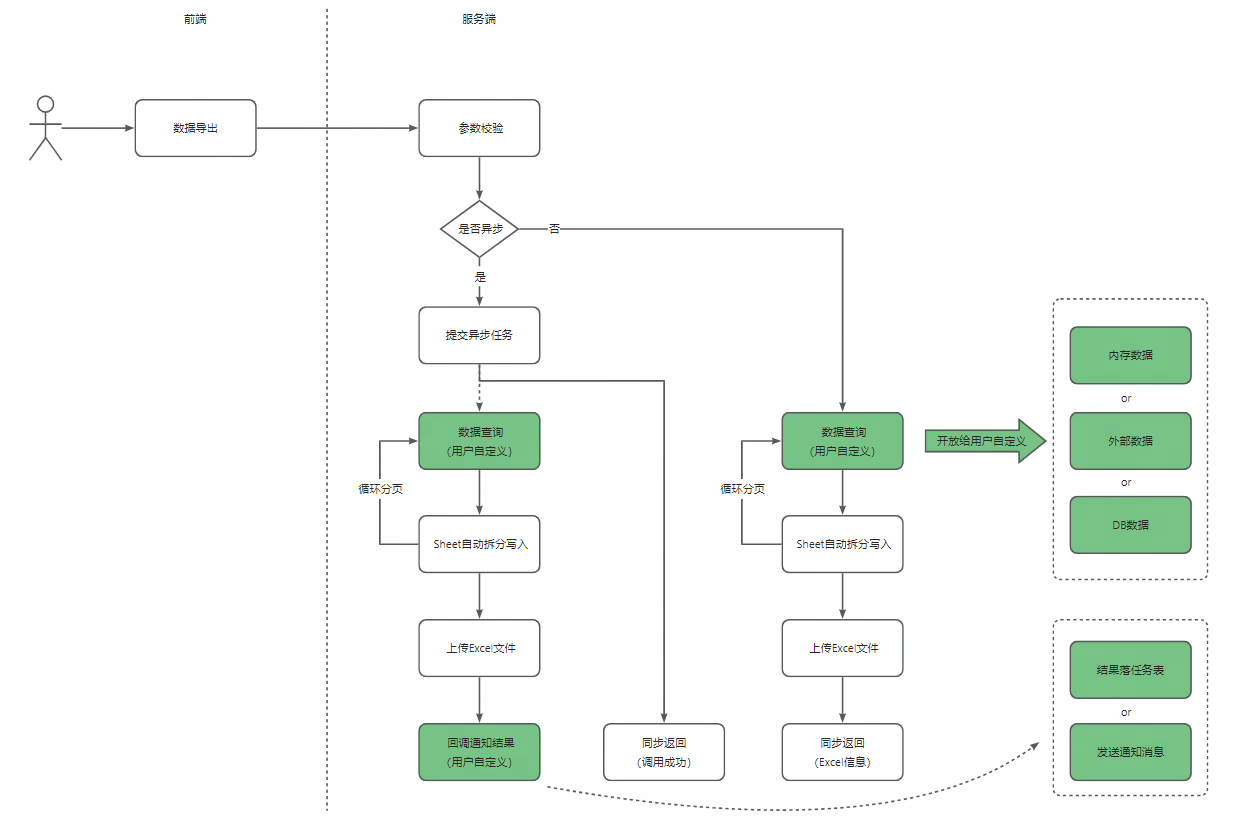
超大数据量处理必然会存在高耗时，解决的核心思想就是异步+拆分。异步就是将原来的导出任务后台执行，避免同步超时；拆分就是将一个超大数据任务拆分成多个小任务。同时拆分后的小任务也可通过多线程来提高处理速率，但是可能存在数据乱序，对有序场景不友好，暂时不做这方面的设计。

**实现原理**

异步：通过线程池将导出任务异步化，避免同步超时；

分页：通过分页将大数据拆分成多个小任务，并允许用户自定义分页数据查询逻辑，以支持更高的自由度；（子任务间串行处理，暂不考虑并行）

回调：由于任务做了异步化，处理结果将通过回调的方式通知用户，并允许用户自定义回调通知处理逻辑。



自定义逻辑说明：

数据查询逻辑：

1.Excel导出逻辑接收一个Function参数，它允许用户添加自定义逻辑。导出任务自动调用该逻辑实现数据查询；

2.在IDE添加逻辑会自动补全Function所需的参数列表，无需手动修改；

3.用户需要按照Function的参数要求，将数据填充到List<RowData>返回。

4.用户可以在自定义逻辑中构造数据、查询DB、调用外部接口。

回调通知逻辑：

1.Excel导出逻辑接收一个Function参数，它允许用户添加自定义逻辑。导出任务为异步模式时，导出完成会自动调用回调

2.在IDE添加逻辑会自动补全Function所需的参数列表，无需手动修改；

3.用户需要处理接收到的任务结果，可以保存在用户自己的任务表，也可以通过其他方式通知相关人；

**性能数据**

硬件：1核2G

通过内存构造3000w数据并导出：耗时504s，导出文件大小452MB。

查询DB构造3000w数据并导出：待测

**使用案例**

如果数据量过大，请调整应用文件上传大小限制：

应用配置->系统参数配置->单个文件上传大小/单次请求大小（默认50MB）

TODO: 后续优化依赖库文件上传方式，不再支持依赖库内直接调用http upload

引入最新版EasyExcel依赖库，exportBigDataExcel逻辑实现了大数据量异步导出功能；

**参数介绍**

入参：

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 说明 |
| queryData | Function<QueryCondition,List<RowData>> | 用户自定义数据查询逻辑 |
| condition | QueryCondition | 数据查询条件 |
| saveResult | Function<ExportBigDataResponse,Boolean>> | 用户自定义回调通知逻辑 |

返回：

|  |  |  |
| --- | --- | --- |
| 参数名 | 类型 | 说明 |
| result | ExportBigDataResponse | 同步返回信息  当设置为同步模式，返回excel导出结果；  当设置为异步模式，返回任务受理信息。 |

QueryCondition

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| pageNum | Integer | 分页查询页码，非负整数 |
| pageSize | Integer | 每页查询数据大小 |
| totalPages | Integer | 数据总页数 |
| isAsync | Boolean | 是否开启异步化  false，同步模式（默认）  true，异步模式 |
| customQueryCondition | Map<String,String> | 用户自定义查询条件 |
| fileName | String | 导出文件名称 |
| sheetTotalRows | Integer | 每个Sheet保存的最大行数 |
| headList | List<HeadData> | 表头信息 |

RowData

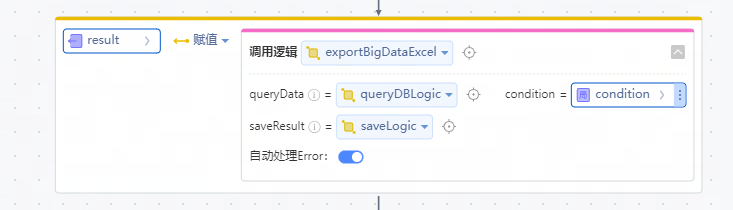
|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| columns | List<String> | 代表一行，list每个item代表一个单元格 |

ExportBigDataResponse

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 说明 |
| success | Boolean | 调用是否成功 |
| msg | String | 描述信息 |
| trace | String | 异常信息 |
| filePath | String | 制品文件下载相对路径  /upload/app/2024010811.xlsx  拼接制品域名可实现文件下载 |
| url | String | 制品文件下载绝对路径 |
| cost | Decimal | 导出任务执行耗时，单位s |
| size | Decimal | 导出excel大小，单位字节 |

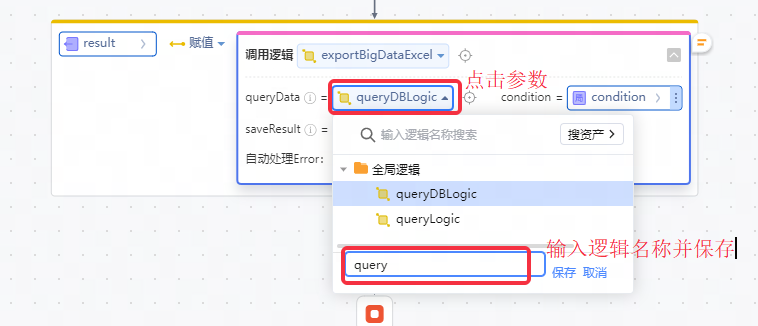
**使用说明**

拖入EasyExcel依赖库的exportBigDataExcel逻辑。

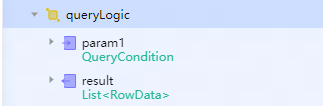


添加数据查询逻辑

这个逻辑用于实际的数据查询

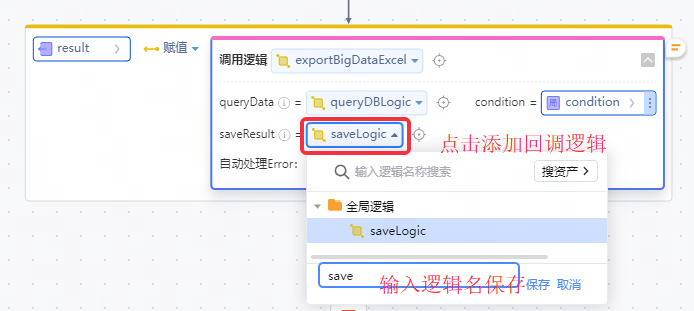


保存后会自动创建逻辑，该逻辑包含一个入参param1和一个返回参数result

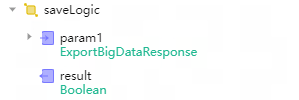


查询逻辑请使用param1中的查询条件，例如pageNum、pageSize等。

添加回调通知逻辑



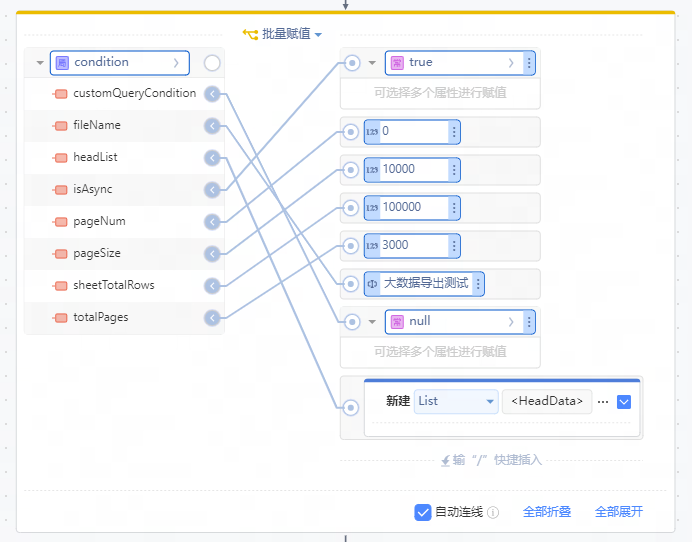
保存后会自动创建该逻辑，逻辑包含一个入参param1和一个返回参数result



入参param1即为异步导出的结果，用户可自定义处理逻辑。

设置查询条件condition

这里给出示例:



customQueryCondition，是一个Map，为用户自定义的其他查询参数，比如userId，在这里设置后，用户可以在queryData逻辑中使用实现真正的数据查询；

fileName，为导出的Excel名称，后缀\*.xlsx。

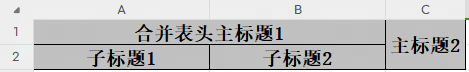
isAsync，是否异步。如果为true，导出任务将同步返回；如果为false，导出任务同步返回任务状态，导出成功后，将调用saveResult逻辑返回任务结果，用户需要在该逻辑中处理结果；

pageNum，设置数据查询的分页页码，查询时请使用该参数；

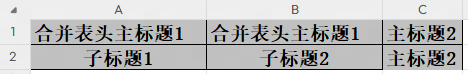
pageSize，设置数据查询的分页大小，查询时请使用该参数；

sheetTotalRows，设置导出Excel的每个Sheet的大小，如果数据量过多将自动拆分Sheet；

headList，是一个List<HeadData>，用于设置表头数据。支持复杂表头，这里给出示例



比如上面的复杂表头，首先我们忽略合并单元格，实际上表头信息是这样的：



A1与B1实际上是由两个相同单元格组成，内容都是“合并表头主标题1”；

C1与C2实际上也是由两个相同单元格组成，内容都是“主标题2”；

按照拆分后的表格，我们设置表头参数：

第一列是A1、A2，那么我们创建一个HeadData，如下



第二列是B1、B2，创建HeadData如下



第三列是C1、C2，创建HeadData如下。



最后我们将三个HeadData依次添加到headList

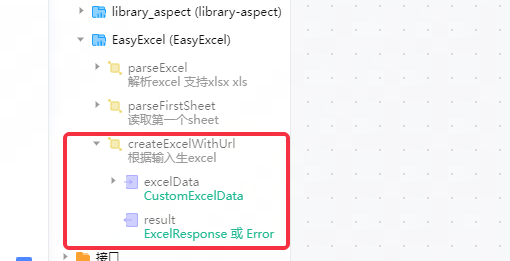


**4. 使用依赖库[推荐数据导出实践v2]**

**4.1 createExcelWithUrl**

根据输入数据生成excel并返回下载链接

查看逻辑入参类型：CustomExcelData



定义入参类型的变量

https://office.netease.com/api/admin/file/download?path=cowork/2023/11/28/f95cfeffc98d4dc6bb19c35d76dccfad.png

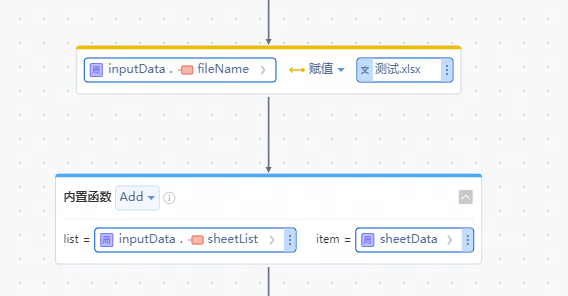
定义excel Sheet

https://office.netease.com/api/admin/file/download?path=cowork/2023/11/28/731ae7bce59342c8bbe7afa83689da0b.png

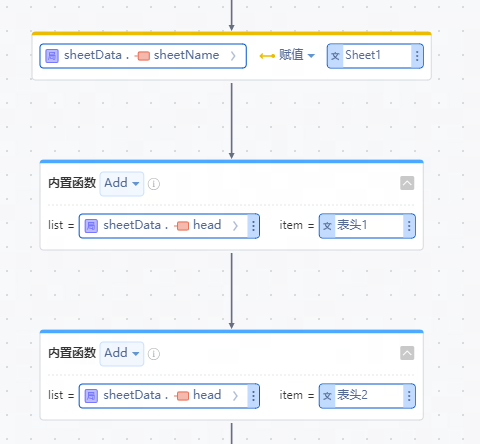
定义数据行

https://office.netease.com/api/admin/file/download?path=cowork/2023/11/28/002fc904e26e4a019f852016e32e02e5.png

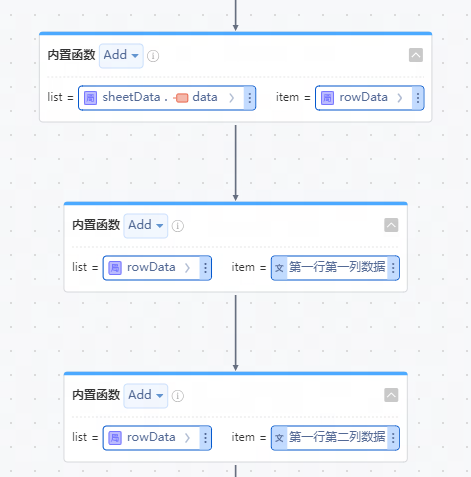
添加文件名，添加第一个Sheet1



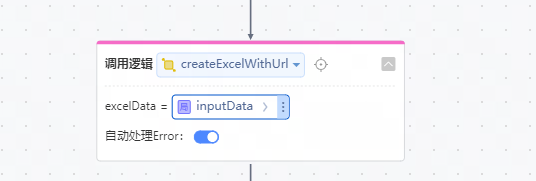
添加Sheet名，添加表头



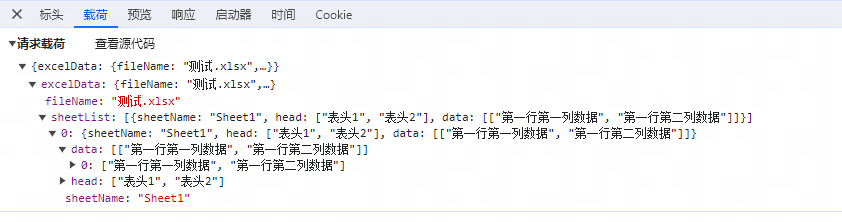
添加数据行



调用依赖库



测试：



如下data字段返回文件下载地址

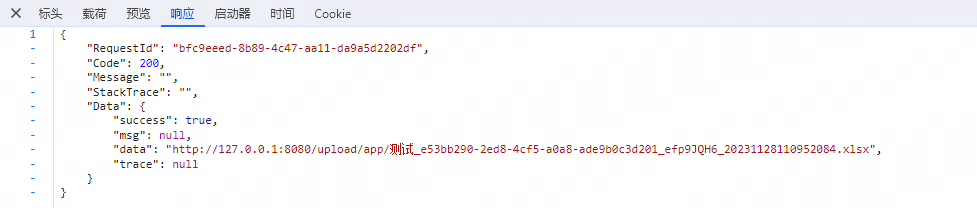
返回的地址：

http://127.0.0.1:8080/upload/app/测试\_e53bb290-2ed8-4cf5-a0a8-ade9b0c3d201\_efp9JQH6\_20231128110952084.xlsx

将http://127.0.0.1:8080替换为制品域名即可直接下载

（这里使用的是local存储因此这里返回的下载链接是127.0.0.1）

https://dev.costtest.defaulttenant.lcap.hatest.163yun.com/upload/app/%E6%B5%8B%E8%AF%95\_e53bb290-2ed8-4cf5-a0a8-ade9b0c3d201\_efp9JQH6\_20231128110952084.xlsx



替换这一步直接使用逻辑将返回结果替换即可。

**4.2 createExcelWithUrl展示图片**

在4.1中，表单数据使用CustomSheetData结构来接收。CustomSheetData中的data字段用于填充表格数据，具体示例请参考4.1。如果图片不需要在单元格中直接展示为图片，可以直接填充图片的URL，例如http://xxxx。如果图片需要在单元格中直接展示为图片，则需要在数据前面添加前缀【codewave\_excel\_pic:】，例如codewave\_excel\_pic:http://xxx。