目录

体验网站说明

写在前面

体验地址：http://47.108.141.193:8031/ 体验网站数据库定期还原，可以随意修改配置查看效果，为方便其他人使用，模板相关的配置请尽量不要动。

体验网站技术清单

- 服务器采用阿里云1核2G，Linux8，Docker部署
- 后端采用的是Abp Vnext，net5，自定义表单部分采用自己封装的一套DDD开发框架，流程引擎采用Elsa
- 前端采用Vue2.X开发，前端框架及控件采用vue ant design实现，前端控件参考网站：https://2x.antdv.com/components/overview/
- 数据库采用Mysql
- 文件存储采用fasfdfs
- 缓存采用Redis+项目内存
- 消息队列采用Rabbitmq
- Job调度采用quartz
- Excel导入导出采用NOPI
- 流程跟踪图采用jsplumb

菜单结构

首页

放置网站统计信息，目前未实现

成员组织管理

部门及其成员管理，常规的部门成员管理

工作台

工作流相关功能，我的代办、已办、流程实例管理、流程定义管理等

消息管理中心

系统统一消息管理中心，包括消息规则设置、消息发送中心、我的消息等

Job执行管理

定时作业管理中心，动态配置定时作业，作业调度情况查看

模版运行时

自定义表单模板也是普通表单，这里配置模板运行的效果

单表_模板

单表管理，最普通的表单，结构为：{单表表单:{单表列表视图:{单表编辑视图}}，包含常规的新增、编辑、删除、批量删除、导入Excel、导出Excel、导出Excel模板、查询、表格分页等基础功能。

单树_模板

单树管理，树结构管理，结构为：{单树表单:{单树列表视图:{单树编辑视图}}，同基础的单表_模板管理，树没有导入导出Excel功能，树自动添加PId字段和Title字段，新增编辑时，下拉选择父节点信息，另外，列表视图为一棵树。

单表报表_模板

普通的表格管理，功能常常用在报表统计，没有新增和编辑功能，只是表格展示数据用，结构为：{单表表单:{单表列表视图}。

1对多2Tab_模板

处理一对多关系，从主表进入，结构为：{主表列表表单:{主表列表视图:{Tab表单:{主表编辑视图,从表列表视图:{从表编辑视图}},主表新增视图}}}，新增主表数据时，只编写主表数据，新增主表数据成功后，直接弹出Tab表单，Tab1为编辑主表数据，Tab2为从表列表信息，从表外键直接关联到了主表id上，编辑时，从表任何操作都直接同步更改到数据库。

1对多2Collapse_模板

处理一对多关系，从主表进入，结构为：{主表列表表单:{主表列表视图:{新增Collapse表单:{主表编辑视图,从表本地列表视图:{从表本地编辑视图}},编辑Collapse表单:{主表编辑视图,从表列表视图:{从表编辑视图}}}}}，新增主表数据时，新增主表数据的同时，可以添加从表数据，从表数据先保存到本地，提交时，主表数据和从表列表数据一起提交到后端批量写入到数据库，编辑数据时，主表数据和从表数据任何修改之后，都直接保存到数据库。

本地单表_模板

单表管理，结构为：{单表表单:{单表本地列表视图:{单表本地编辑视图}}，新增、修改、删除数据均在本地，不会提交到后端，做为批量新增数据的基础表单

单树Tree_模板

单树管理，同“单树_模板”，只是树数据展示为树控件，常常用于一对多一为树的场景

单树TreeItem_模板

一对多一为树的场景，布局左边为可编辑树，右边为子表列表视图，点击树节点，展示树对应的子表数据

模板测试

根据模板生成的测试模块

表单常规示例

特殊场景的应用示例，特殊场景的一些功能支持，自定义表单配置重要参考

控件综合应用

各种自定义表单高级用法示例，编辑视图字段自定义验证、字典单选、字典多选、自动编号（雪花算法、流水号自增）、Bool字段处理、关联外键表查询单选、关联外键表查询多选择、关联外键树单选、关联外键树多选择、用户单选、用户多选择、外键关联表字段冗余。列表视图自定义列显示、列表按钮操作权限、普通查询和高级查询各种控件，包括：日期范围、字典下拉、字典瓦片查询、用户选择查询、外键表搜索查询、外键表树查询，查询都支持单选和多选查询。

出入库录入管理和库存盘点管理

常规的表单和列表管理，主要演示二次开发自定义后端方法调用，先进行库存录入，录入完成之后，调用后端自定义方法，完成库存盘点归档操作。

合同管理流程和合同归档流程

综合流程管理演示，父子流程，父流程为合同管理，子流程为合同归档流程，父流程流转到子流程活动时，自动发起合同归档流程，合同归档的合同编号和合同金额从合同管理表单带入到合同归档流程

一对一

一对一关系处理，列表包含两张表的字段，主要演示列表视图关联查询支持。

流程表单运行时

流程相关模板演示

流程表单测试

根据模板生成的流程相关的应用模块

用户体验测试

体验生成的功能模块请配置到此处

表单定义管理

自定义表单所有配置都在这里

Object管理

对象管理，见对象管理相关文章和wiki描述

View管理

View管理，见View管理相关文章和wiki描述

Form管理

Form管理，见Form管理相关文章和wiki描述

系统设置

系统角色管理

功能模块管理

数据字典管理

框架基础的一些数据字典，不在自定义表单处使用

自定义表单数据字典

自定义表单使用的数据字典

00-写在前面

- 前端采用Vue2.X开发，前端框架及控件采用vue ant design实现（后续会支持移动端的自定义功能，后端不需要更改，前端还是采用Vue，可能会选择uniapp）。后续很多设置都是围绕着前端控件来设置的。，前端控件参考网站（因为开始做的时候，VUE3还不成熟，前端后续会升级成Vue3版本）：https://2x.antdv.com/components/overview/
- 附录章节里面的内容很重要，文档里面写到参见附录的地方需要额外关注。
- 所有的配置都是以驼峰命名，如果对象配置里面不是采用驼峰命名，没有特殊说明的地方，都以此标准。如对象属性配置为UserName，其他配置的地方，都为userName。
- 绝大多数逻辑都是围绕动态json处理，保证设计上的灵活性，前端和后端都是。
- 注意Object对象命名和Object对象字段名称不要使用数据库系统关键字，后续执行默认方法时，系统会对参数进行Sql注入判断。
- 外键命名统一规范：对象名称_Id
- wiki系列文章主要是针对自定义表单说明，偏向于技术方面，有些地方不一定好理解，需要结合体验网站项目理解，多数都能够在体验网站找到案例
- 所有内容全是干货，设计上绝对独具一格，你在其他地方很难看到，拒绝啰嗦

01-对象管理

主要配置对象、属性、方法，对象映射为数据库表，属性映射为数据库字段。

设计参考：

- 企业级自定义表单引擎解决方案（三）--实体对象模型设计 - 企业级自定义表单引擎解决方案（四）--实体对象模型实现

对象配置

- Object名称 对象名称、同时也是自动生成的数据库表名称，后续执行的默认Sql方法时，选择的对象也是此值，对象名称不能包含Sql关键字，自定义表单执行后端方法时，会验证所有输入是否包含Sql关键字。
- 应用 对于比较大的应用，可以对一些高度内聚的功能划分为不同的应用，相当于对整个自定义表单的一个分类管理。
- 联合索引 唯一定位一条数据，如果设置了，在新增、修改、导入时，会验证联合索引在数据库是否唯一，因为存在删除审计功能，所以不是在数据库增加唯一索引，而是在业务逻辑层验证唯一性。
- 远程方法设置（已废弃，用通用的后端方法调用代替） 应用场景主要是在子表选择主表数据关联时，构造下拉框带搜索功能的下拉数据的方式，每个Object只有一个，如：Object名称为User,设置值为：`{"script":" ${obj.string1} (${obj.sex}) ","fields":"Id;String1;Sex","whereFields":"String1;Sex","orderInfo":"String1 desc","resultCount":20}`，则生成的Sql语句为：SELECT Id;String1;Sex FROM User WHERE String1 LIKE '%查询条件%' OR Sex LIKE '%查询条件%' ORDER BY String1 desc LIMIT 20; 返回List集合数据(包含value和text属性)，其他value为数据库Id字段值，text为String1(Sex)解析出来的值。
- 主键类型 Int或者Guid，Int时，数据库维护自增Id，新增时，生成Guid做为主键，为统一管理，每个对象都必须有一个主键，且名称为Id，系统不支持联合字段做为主键
- 是否流程表单 主要是表单需要挂接到流程引擎时使用，勾选时，系统自动增加并维护InstanceId(流程示例Id),FlowStartTime(流程发起时间),FlowStartUserId(流程发起人)字段（流程状态、流程结束时间、流程当前活动目前采用调用工作流方法动态实时读取，后续可能会采用实时写入业务表方式+消息队列补偿的方式同步更新到业务表）
- 是否为树 自动添加并维护PId、TreeCode、Path、Icon、Title字段
- 新增审计 自动增加并维护CreatorId(创建人Id),CreationTime(创建时间),CreatorDept(创建者所在部门Id)字段
- 编辑审计 自动增加并维护LastModifierId(最后修改人Id),LastModificationTime(最后修改时间)字段
- 删除审计 当执行删除时，系统不直接删除数据，而是维护一个IsDeleted字段，查询时会根据此字段先过滤已经标识删除的数据，另外还会维护DeleterId(删除人Id)，DeletionTime(删除时间字段)

属性管理

根据模板生成自定义表单时，编辑视图会自动根据这里的属性生成编辑视图对应的属性列，自动增加选择默认的控件以及增加默认的验证，列表视图自动生成表格列以及特定列的格式化显示，自动生成导入导出的Excel模版及验证。

- 属性名称 数据库字段名称，注意：属性名称如果填写的是大写的英文字段，在后面视图和表单应用的地方，都是转化为驼峰命名后的字符串使用，如UserName，则其他地方用的时候为userName使用。
- 描述 不能随意填写，根据模板生成自定义表单时，会根据此字段生成编辑视图的字段Label名称和列表视图的表格标题。
- 字段类型 包括：String、Int、Date、DateTime、AutoNumber、AutoGuid、Guid、Text、Float、Decimal、AutoNo，AutoNumber是指数据库Int自增，AutoNo（本质上为String）指自动生成的编码（默认为雪花算法，另外还可以配置使用自增的流程号方式，如：TS20220627000001等这种格式，后续实现细节会单独写文章描述），注意：没有Bool字段类型，Bool字段类型用字典boolDict代替
- 是否可空 数据库映射的字段会增加IS NOT NULL
- 是否必填 模板生成的编辑视图自动增加必填验证
- 是否唯一 数据库映射自动增加字段唯一索引
- 长度 String、Text、AutoNo为字符长度，数据库映射的字段长度，如Varchar(长度)，模板生成的编辑视图会自动添加字符长度验证，Decimal为整数部分长度
- 长度2 Decimal填写，数据为映射为Decimal(长度,长度2)
- 默认值 同数据库设计的默认值功能相同，在采用雪花算法自动生成编号的场景中，这值为雪花算法生成的字符前缀。
- 排序 模板自动生成的编辑视图、列表视图等的字段顺序
- 默认控件 模板生成的编辑视图默认使用的控件，系统已经添加和最适合的控件，一般情况下不需要更改
- 默认控件属性 配合默认控件属性使用。
- 附加验证 在模板生成的编辑视图系统根据属性配置，已经添加了默认的验证，对于一些特殊的验证，可以在此处添加验证规则。
- 字典 数据类型必须选择String，本质上还是String类型，数据库存储字典编码，在模板生成的编辑视图、列表视图查询、列表视图列等地方均用下拉选择数据字典控件代替。
- 字段配置 特殊引用场景增加字段的自定义配置，如在选择AutoNo生成流程号的场景，配置自动生成的流水号规则，如：配置淡`{"serialNumber":{"preNo":"ST","length":6,"useDay":true,"useMonth":true,"useYear":true}}`，生成字段如ST20220617000001

方法管理

自定义表单将大多数应用系统的常规业务场景都封装好了，绝大多数场景不需要自定义方法调用，但对于特殊的应用场景，比如复杂报表统计、复杂后台业务逻辑，这些场景提供了自定义方法的功能（二次开发接入），在表单规则引擎调用后端方法时，选择此方法Id即可。

- 方法名称、方法描述 没有实际的用处，只是在这里显示出来而已供表单配置者自己查看
- 执行类型 包括Sql、反射、微服务、API，目前只支持反射，但也足以使用，后续会陆续增加API、Sql执行，再后面会支持微服务，微服务只能采用自己封装的微服务组件才可以
- 执行内容 反射为类程序集名称。如：CK.Sprite.Form.Business.StockDomainService, CK.Sprite.Form.Business，自定义的类需要实现IBusinessExec接口。使用请参考源码项目CK.Sprite.Form.Business，请参考进销存相关示例。 对于其他字段可以不管，后续其他调用方式支持时，再完善此部分文档。

02-视图管理

写在前面

- 定义一个视图时，每个字段都建议填写，且每个字段的意义都需要明确的代表你想要表达的意思，方便其他地方定义选择视图时一眼就看得清楚。
- 视图定义为业务逻辑最小化单元，视图内部高度内聚，视图只了解自身和子表单和子视图的信息，对父表单或者父视图一无所知，对孙子表单或者孙子视图也是一无所知的，如果要控制操作孙子表单或者孙子视图，则需要设置子表单或者子视图规则，子表单或者子视图再设置规则操作他的子表单或者子视图。

设计参考：

企业级自定义表单引擎解决方案（七）--视图模型管理

公共字段

- 视图名称 与具体的业务逻辑没多大关系，只是显示用，具体的业务与视图相关的都是选择视图的Id使用。
- 视图类型 分为ListView（传统Table列表），ItemView（传统新增编辑表单），TreeView（树），ListLocalView（本地Table列表，增删改不会立即提交到后台，完成后统一提交到后台保存数据，常用在新增主表数据同时新增子表集合数据场景），目前只支持此四种视图，后续会扩展支持其他类型的视图。
- 描述 详细描述视图的用途
- 业务分类 相同功能的业务命名相同，与表单处的业务分类保持一致，方便查询时快速找到想要的视图或表单，与实际业务无关。
- 对象 视图里面的业务涉及到哪些对象就把对象名称填写到此处，多个用英文分号;分割，如：Contract或者OTN1;OTN2。
- 属性 Json字符串，具体的属性配置与视图类型相关，见具体视图描述。

列表视图（ListView）

列表视图极为传统的表格管理，表格包含表格内容、查询、高级查询、分页、操作按钮、列格式化显示、行按钮等的管理 列表视图常规数据渲染过程：*定义页码改变事件，事件执行的规则为调用后端的PageList方法，参数从页码控件、隐藏参数查询、普通查询、高级查询、表格表头排序动态拼接，组装成查询参数，然后再执行绑定数据规则（获取参数过程列表视图已经封装，直接配置使用即可），将查询结果绑定到ant表格控件。另外，在视图加载、查询按钮点击、高级查询查询、点击排序表头、弹出子表单或子视图对话框确认事件中，都会直接绑定到执行页码改变事件，进而执行数据渲染过程。（主要是用规则引擎将所有动作串联起来，规则引擎参见规则引擎部分文档）*。单表的数据一般比较好设置，多表关联时，需要定义别名以及关联Sql语句等，如一对一关系配置执行后端方法配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"OTO1","methodId":"MultiPageList","transfors":[{"transType":"query","aliasInfos":"OTO1:a;OTO2:b","joinInfos":"OTO1 a LEFT JOIN OTO2 b ON a.Id=b.Id"}]}]}`，多表查询映射出来的数据每个字段都会增加“_别名”处理，如id_a,userName_b等。

属性设置

属性常规设置即为设置ant的table控件，另外附加扩展的一些自定义设置，ant的Table设置参考Table组件设置，其他一些扩展字段如下：

1. tableDiv 设置a-table外层的div样式，默认样式为：'min-height: 560px'
2. rowKey 前端表格的行主键，一般情况为"id"，可以不设置，当查询出来的列表数据主键不为id是，需要设置，如：id_a
3. tableType 自定义扩展字段，多数情况不需要设置，如果是树列表，需要设置为“tree”
4. columns 参见Table组件设置，定义表格列
5. colOperateWidth 定义列表操作列的宽度，当列表操作控件或者列表更多控件定义了，列表中的操作列才会显示。
6. excelTemplate 导入导出Excel配置模版信息，参见附录【导入导出Excel模版】，如：`[{"name":"电话","field":"phoneNumber","fieldType":5,"isRequred":true,"validateType":99,"validateValue":"[{'customerVal':'Phone','trigger':'blur'}]"},{"name":"学历","field":"education","fieldType":5,"validateType":11}]`
7. excelDicts 字段字典映射集合数据，表示Excel中用到的数据字典映射信息，定义哪个字段用到哪个数据字典，如：`[{"dict":"sex","field":"sex"},{"dict":"edu","field":"education"},{"dict":"title","field":"positionalTitle"}]`
8. excelName 导入导出Excel文件名称，如：人员管理测试Excel数据
9. uniqKey 导入时唯一字段检测（这里的字段为数据库字段，注意不是转换为驼峰命名之后的字段，多个字段组合用;号隔开），如：UserName。
10. eval_query 执行后端方法获取查询参数后执行的JS脚本，自定义扩展处理查询条件，本质上执行eval函数，特殊场景使用，比如执行查询之前，将查询条件做自定义特殊处理，如：界面查询条件只查询年月，到后端映射为时间段查询`sqlWhere.children.forEach(r=>{if(r.field==='checkTime'){r.value=[r.value.format('yyyy-MM')+'-01 00:00:00',r.value.add(1,'month').format('yyyy-MM')+'-01 00:00:00']}})`，参考附录：【自定义查询条件】
11. defaultSorting 默认后端查询方法排序，如：`checkTime desc,stockCheckType asc`等

列管理

对列表区域特定列数据的格式化显示

- 绑定字段 查询出来的数据特定字段

- 组件名称 col-tbl-text或者col-tbl-icon
- 编辑组件 未实现此功能，原本定义为列表视图可直接在表格中编辑，现取消此功能。
- 组件属性设置

1. 格式化显示特定行列数据，可使用的上下文参数：text（特定行特定列的原始文本）、record（特定行的整行数据对象），index（行索引）
2. col-tbl-text控件主要是针对文本格式的转换，动态调用js的eval方法，定义eval_render值，内容为动态执行的js语句，如：`{"eval_render":"text.substring(0, 19).replace('T', ' ')"}`，将时间字段格式化显示。
3. col-tbl-icon控件主要是针对字典或者需要定义图标显示的列，主要填写内容：icons(图标格式显示)、texts(文本格式显示)、dict(数据字典名称)、isMulti(数据字典时使用，是否多选择，数据字典多选时，不填写icons，构造出"字典项1;字典项2"的内容)、showTitle(是否显示文本)，icons的配置参考前端网站a-icon的配置,texts配置为span的配置。),showTitle(是否显示文本)几个字段，icons的配置参考前端网站a-icon的配置,texts配置为span的配置。
4. 例1（只显示图标，不显示文本，其中key为字段值，config为根据特定值做特定的配置）：`{"icons":[{"key":"female","config":{"type":"woman","style":"font-size:14px;font-weight:600;color:rgb(255, 112, 112);","title":"女"}},{"key":"male","config":{"type":"man","style":"font-size:14px;font-weight:600;color:#13c2c2;","title":"男"}}],"showTitle":false}`
5. 例2（同时显示图标和文本，key为空，标识匹配任意文本）：`{"icons":[{"key":"","config":{"type":"phone","style":"font-size:14px;font-weight:600;color:#13c2c2;"}}],"texts":[{"key":"","config":{"style":"font-size:14px;font-weight:600;color:#13c2c2;"}}],"showTitle":true}`
6. 例3（数据字典，不带图标，格式化显示）：`{"texts":[{"key":"master","config":{"style":"font-size: 14px;font-weight: 400;color: rgb(19 113 194);"}},{"key":"doctor","config":{"style":"font-size: 14px;font-weight: 400;color: rgb(70 194 19);"}},{"key":"","config":{"style":"font-size: 14px;font-weight: 400;color: rgb(194 189 19);"}}],"dict":"edu","showTitle":true}` 特殊说明：如果通过模板自动生成，如果字段类型是bool,date,datetime，或者定义了dict，系统会自动添加格式化列显示。

查询（查询涉及到各种控件的使用，参见附录【自定义表单控件使用说明】描述）

查询主要定义执行后端方法PageList或者ListWhere前，根据用户输入的查询条件，动态构造查询树对象，参见附录【自定义查询条件】

- 查询类型 普通查询(列表查询区域看到的查询控件)、高级查询(点击查询区域最右侧“更多”按钮显示的高级查询功能)、隐藏查询(用户看不到，通过规则引擎的方式给列表查询赋值)
- 查询字段 字段名称
- 允许查询的条件 普通查询和隐藏查询条件只填写一个，与后面的默认条件一致，高级查询条件可以选择多个供用户选择使用，选择查询条件时，需要适配使用的控件以及字段类型，比如a-input控件选择查询条件为In，这种在运行时会报错。
- 默认查询条件 单选
- 查询源（已废弃） 配合组件使用，调用远程方法，获取组件需要的数据源，数据源统计返回带name和value字段的集合。
- 组件名称：

1. 普通查询见附录【自定义表单控件使用说明】描述
2. 高级查询 高级查询控件的使用方式与普通查询比较类型，请参见 > 体验网站> 控件综合应用

> TextBox，如果查询条件为Between，则构造两个文本框输入，否则构造一个文本框输入 SingleDropDown和MultipleDropDown，下拉单选或者下拉多选 TreeSingleDropDown和TreeMultipleDropDown，下拉单选或者下拉多选树 SingleDropDownAndTextBox和MultipleDropDownAndTextBox，下拉单选或者下拉多选，允许用户输入查询条件搜索 SingleTile和Radio，瓦片和Radio单选 DateTime，如果查询条件为Between，则构造日期范围选择控件，否则构造日期控件。 col-form-user，系统用户查询选择控件 下拉多选的地方，查询条件多半为In或者NotIN

编辑视图（ItemView）

- 行管理 比较好理解，比较简单，只需要定义排序集合，也可以定义行的Div样式设置。
- 列管理

> 视图行：列所在的行 列类型：Control，View或者Form，当选择View或者Form时，在此列渲染子表单或者子视图 对象Id：如果列类型为View或者Form，设置为View的Id或者Form的Id 验证：如果是Control，可以定义验证规则，参见附录【表单验证】，如：`[ { "required": true, "message": "请填写此项数据", "trigger": "blur" }]` 绑定值：如果是Control，绑定到的object对象字段 属性：这里是ant-design的Item属性设置，不是控件的属性设置 Wrap信息：参见【公共配置Wrap管理】 Label显示：Lable显示 Label属性：一般不填写，设置Label的属性 控件和控件设置：见附录【自定义表单控件使用说明】描述

树视图（TreeView）

同列表视图配置基本相同，只是规则执行后端方法时，执行的是TreeListWhere等树相关的方法，另外界面的一些控件就没有了，比如分页，导入导出Excel等。基本的列表视图理解之后，参考 > 示例网站 > 模板运行时 > 单树Tree_模板

本地列表视图（ListLocalView）

同列表视图配置基本相同，只是新增、修改、删除都是针对本地表格数据进行的操作，不执行后端方法，参见 > 示例网站 > 模板运行时 > 本地单表_模板，以及参考 > 示例网站 > 模板运行时 > 1对多2Collapse_模板

Wrap管理和控件管理

参见【04-公共配置管理】

03-表单管理

定义一个表单时，每个字段都建议填写，且每个字段的意义都需要明确的代表你想要表达的意思，方便其他定义选择表单时一眼就看得清楚 表单定义为视图容器，视图与视图之间是彼此不知道的，通过规则将表单视图串联起来，构造整体的页面，表单也是从菜单进行业务功能模块的路径。一个表单对应多个表单项，一个表单项对应多个表单行、一个表单行对应多个表单列、每个表单列可以添加包裹的子视图或者子表单 表单模型实际定义为一个容器，容器里面会进一步定义行列，容器里面可以包含表单或者视图，视图和表单里面又有各种对象，比如按钮、Wrap信息等，每一个页面会定义唯一一个最外层的表单容器，我们可以把它看作根容器，这样就整体形成了一棵树，根节点就是最外层的表单定义，树的节点可以是子表单、视图、表单行、表单列、视图行、视图列、视图控件等，整体就可以构造出一树庞大的树。每一个节点都会有一个节点Code，做为页面唯一对象的定位，这样规则引擎就能够精准的定位到想要控制的对象中。

设计参考

企业级自定义表单引擎解决方案（八）--表单模型管理

表单属性

- 表单名称 显示名称，与业务无关，请填写有意思的表单名称
- 业务分类 相同功能的业务命名相同，与视图处的业务分类保持一致，方便查询时快速找到想要的视图或表单。
- 表单类型：NormalForm、TabForm、DivForm、StepForm、CollapseForm，后续还可能增加其他的表单类型，不同的表单类型定义不同的容器
- 描述 表单描述信息
- 属性设置 针对不同表单类型的属性设置
- 菜单进入配置 对最外层表单的Wrap配置，参见【Wrap管理】

表单项

表单行的容器，表单项为一个Div容器，一个表单可以定义多个表单项，表单为表单项的容器，不同的表单类型构造不同的最外层容器，比如TabForm构造tabs标签的容器，每个tab页为一个表单项，其他表单类型类似。

表单行

同编辑视图的视图行定义，只是对表单列的布局进行管理

表单列

同编辑视图的视图列定义，表单列常常用在添加子表单或者子视图功能用，当然也可以添加普通控件，比如按钮等。参考【编辑视图行列管理】配置列类型：View、Form或Control 对象Id：View或者Form的Id

Wrap管理和控件管理

参见【04-公共配置管理】

04-公共配置管理

Wrap管理

Wrap应用在对子表单或者子视图进行外包装处理，可以用模态对话框、抽屉、Div等进行包装处理，比如在用户管理列表视图(userListView)打开用户编辑视图(userItemView)时，在用户管理列表视图定义Wrap信息，将用户编辑视图包装在Model模态对话框里面，在点击新增按钮触发显示模态对话框。再如在部门用户管理中，在最外层的表单管理中，定义两个表单列数据，一列显示部门树管理视图，一列显示用户列表视图，则可以将用户列表视图用Card卡片包装起来。视图和表单对本身在何处使用可以是一无所知，在使用的地方，在使用的地方，就可以用Wrap进行自己想要的包装，比如增加Div样式、模态对话框弹出、Card样式封装等。 Wrap可以用在表单或者视图的Wrap信息管理和表单列或者编辑视图的列信息进行Wrap定义。Wrap支持嵌套处理，比如对子表单可以先进行Div包装，再进行Card包装，最后还可以进行模态对话框包装。 Wrap对象会加入到表单或者视图的对象管理链中，封装的对象Id为“包裹对象_Wrap配置id”，在自定义表单规则处就可以直接使用。 业务分类 自动添加，标识是视图或者表单的Wrap

- 对象id 即包装的子表单或者子视图Id（不是视图名称，是视图的数据的id字段的Guid值）
- 组件名称 子表单为form-layout，子视图为view-layout，必须对应选择填写
- Wrap配置举例

```
[
  {
 "id": "drawer1",
 "componentName": "drawer-wrap",
 "wrapSettings": {
   "visible": false,
   "title": "单表信息管理",
   "width": 1024,
   "isEnableButton": true,
   "maskClosable": false
 }
  },
  {
 "id": "div1",
 "componentName": "div-wrap",
 "wrapSettings": {
   "style": {
  "background": "#eceff1"
   }
 }
  }
]
```

- 例子说明

1. 表示先对子视图最外层先进行抽屉效果Wrap封装，再抽屉里面再进行div封装。
2. id：定义包装标识，后面规则定义时会选择到此id，比如点击按钮打开抽屉，其实就是找到此id的抽屉，将visible设置为true。
3. componentName：包装组件，可以选择的值：div-wrap(div封装),modal-wrap(a-modal模态对话框封装),drawer-wrap(a-drawer抽屉效果封装，自定义的属性设置isEnableButton，标识是否显示底部按钮),spin-wrap(a-spin等待效果封装),card-wrap(a-card卡片效果封装)
4. wrapSettings：包装控件属性设置，具体属性设置值参考前端网站，drawer-wrap(增加自定义的属性设置isEnableButton，标识是否显示底部按钮)
5. 其他说明：modal-wrap和drawer-wrap点击关闭按钮和确定按钮会发出cancel和ok事件，可以规则定义处检测处理

控件

特定的视图或者表单在预留特定的区域方式操作控件，配合规则引擎完成业务功能，比如列表视图的树视图的控件管理。 控件会加入到表单或者视图的对象管理链中，对象Id为即为控件的id字段Guid值，在自定义表单规则处就可以直接使用。

- 业务分类 view或者form，目前只需在ListView处定义。
- 操作类型

1. ListView视图方法：针对整个ListView或者TreeView的操作，比如新增、导出Excel等操作按钮区域
2. ListView查询按钮：查询或者清空按钮区域。
3. ListView操作方法执行：针对每行数据的常规操作，比如编辑、删除、查询等。
4. ListView列表更多方法执行：第对每行数据的操作，显示在更多下拉选择区域。

- 排序 针对某个区域按钮的显示先后顺序排序。
- 描述 只是列表显示用，和业务无关。

- 组件名称 ListView视图方法和ListView查询按钮使用col-button，ListView操作方法执行和ListView列表更多方法执行使用col-tbl-button
- 组件属性 如：`{"name":"新增","type":"primary","icon":"plus"}或者{"name":"编辑","type":"link","icon":"edit"}`，参见配置参考网站的按钮部分内容配置。

05-自定义表单规则引擎

规则引擎是自定义表单最为核心的内容，也是与其他低代码工具最为本质的区别，非常重要的内容，对于读者来说也是最难理解内容。 没有规则引擎的低代码工具，即使前端做得再花哨，也仅仅是解决了一些复制粘贴的问题 视图和表单都可以定义规则，规则将所有业务串联起来，至关重要。

对象树

进行自定义表单页面，都会定义唯一一个最外层的表单，表单里面可以定义子表单、子视图（视图里面也可以定义子表单和子视图），视图或者表单里面会定义各种对象（可以是控件、弹框、行列信息等），整体构造一棵树，每一个节点都有唯一的编码，编码是根据树的每一个节点的Id组成，通过编码就可以找到唯一的对象。比如界面上的新增按钮的编码为“表单Id.列表视图Id.新增按钮控件Id”。 视图和表单Wrap、表单的表单列、ItemView的列，这几个地方可以定义子表单和子视图。

规则管理

表单、视图、控件都会触发各种事件，表单或视图对父表单或父视图一无所知，且只能控制操作直接子表单或子视图的对象，事件规则查找逻辑为：

1. 如果是表单触发的事件，系统会查找本表单触发的规则（子表单Id、子视图Id、对象Id都为空）和父级（父表单或者父视图）定义的规则（子表单Id为触发事件的表单Id，子视图Id为空，对象Id为空）
2. 如果是视图触发的事件，系统会查找本视图触发的规则（子表单Id、子视图Id、对象Id都为空）和父级（父表单或者父视图）定义的规则（子表单Id为空，子视图Id为触发事件的视图Id，对象Id为空）以及当父级为表单时，查找父级的父级定义的规则（子表单Id为父级表单Id，子视图为触发事件的视图Id，对象Id为空）
3. 如果是对象触发的事件，系统直接查找父级规则（子表单Id为空、子视图Id为空、对象Id为对象Id）和父级的父级规则（子表单Id为空或者子视图Id为空（取决于父级的类型）、子表单Id为空或者子视图Id为父级Id（取决于父级的类型）、对象Id为对象Id）和父级的父级的父级（触发事件的对象的父级为视图，父级的父级为表单，且还会继续网上查找一级的规则，查找逻辑与其他情况类似） 上述描述不好理解，可以结合到体验网站各种配置好的表单查看，多看一些规则就能理解清楚。

- 业务分类 只能是View或Form，系统自动选择
- 子表单Id 子表单Id
- 子视图Id 子视图Id
- 对象Id 表单或者视图内不同控件的Id，注意：Wrap对象的Id为：Wrap包裹的表单或视图Id_Wrap配置id
- 事件名称 基础控件的使用参见不同控件的ant的事件，自定义表单会将所有ant控件的事件抛出，可以定义规则进行拦截，其他自定义事件参考如下

> viewloaded：视图加载时触发的事件 selectedrows：ListView的选中行 advancequery：ListView的高级查询按钮点击事件 pagechange：ListView分页触发事件 refresh：ListView页面刷新事件 selected：树视图节点选择事件 Workflow_FormValidate：工作流界面触发表单验证事件（流程管理界面点击提交按钮等地方） Workflow_GetMethods：工作流管理获取表单信息事件（工作流管理界面装载自定义表单时，需要获取自定义表单信息） Workflow_FormOpen：打开工作流管理界面事件 Workflow_TrySend：工作流预提交完成 Workflow_Save：工作流保存执行完成 Workflow_Send：工作流提交完成 另外，还可以自己定义一些事件，并配置执行规则，这些事件不是由系统自动触发，而是在需要的时候，可以在其他规则执行中选择执行其他规则，事件名称选择自定义的事件名称即可，这样就可以触发执行自定义的事件规则。比如在列表视图中，多个地方都会触发查询列表数据并绑定到表格中，此时就可以定义一个自定义的事件及规则，其他地方直接使用即可。

规则执行管理

一个事件会触发一系列的规则执行，每个规则执行都会有一个规则执行所在的宿主对象，宿主对象为事件所属的视图或者表单，规则执行都在相对宿主对象来定义执行的。 子表单、子视图、对象Id：这几个字段都是相对宿主对象来说有，某些规则执行需要指定规则执行的目标对象，比如绑定数据，需要知道将数据绑定到哪个目标对象上，比如可以绑定到本视图的某个按钮或者子表单的子视图的某个对象上。

- 子表单Id 子表单Id
- 子视图Id 子视图Id
- 对象Id
- 排序 定义规则执行的先后顺序
- 执行类型和执行配置 不同的执行类型定义不同的配置支持，具体规则如下：

1. 设置属性(actionType=1) 设置目标对象的自定义属性，比如点击新增按钮弹出对话框，参考前端组件的属性，则设置对象为对话框Id（参见【Wrap对象定义】）：66e68bf9-8e05-4691-bf99-ed95c820260d_modal1，执行设置为：{"visible":true} 特殊属性设置

> 弹出导入Excel对话框，{"type":"importexcel","objectName":"TemplateTests","applicationCode":"Default"} 设置页码为1，{"skipCount":0} 弹出模态对话框，{"visible":true,"bodyStyle":{"padding":"10px"}}，对象Id为af1756bb-da48-4b5f-821d-78aeeaddbd2e_modal1

2. 执行方法(actionType=2) 执行后端服务器方法，自定义表单默认了一系列默认方法执行，默认情况下满足绝大多数业务场景，默认方法不满足的情况下，可以定义Object对象方法，自定义执行逻辑，自定义方法的执行参考【方法管理】 方法执行配置结构：

> applicationCode：应用编码，默认即可 isTransaction：是否封装到事务中执行，多数情况填写true methods：执行的一系列后端方法集合，具体参数如下：

> - ruleId：方法执行Id，任意定义的值，用于存储方法执行的结果
> - objectName：方法所属的对象
> - methodId：如果是执行自定义的Object对象方法，则填写Object对象方法的Id，否则填写系统默认的方法名称，具体名称见下文
> - paramModel：server或者local，默认为local，定义方法参数是从前端绑定还是后端其他方法执行结果获取
> - serverParams：如果paramModel为server时使用，
> - userFields：定义查询结果哪些字段是需要构造用户名称的字段（数据库只存储用户id，界面显示的时候，需要显示用户名，系统自动为查询结果添加"原始字段_UserName"字段）
> - remoteFieldInfos：远程关联字段特殊处理（常常用在外键关联的地方，数据库只存储远程对象的Id，界面需要显示外键绑定的值），具体参考【外键关联处理】
> - instanceIdField：如果表单挂接到流程引擎，系统自动带出流程相关数据字段，这里定义关联的流程定义字段名称，系统查询流程管理，获取流程审批状态等信息
> - execType：workflow_formId或者workflow_fact，只在流程引擎处使用，标识方法是获取表单Id值还是事实库Fact值
> - baseConfig: 基础配置，可空，配置之后，会先应用基础配置，再执行后续的传递替换参数值
> - transfors：执行方法所需的参数的绑定规则，真正请求的时候，参数封装到datas对象（如果是数组，请采用"字段名称:数组索引"，具体设置某个数组对象的值），datas具体属性内容参见不同方法执行配置举例。传递数据配置：
>
> 1. paramName：需要赋值的参数名
> 2. transType：参数传递类型，query、exportexcel、eventData、userInfo等，具体值参考后续规则执行示例，userInfo表示当前登录用户信息，eventData表示从事件触发处获取参数值，比如点击列表删除按钮，会将当前删除按钮所在行的数据加入到触发事件的事件绑定值上，可以在规则执行时获取
> 3. transTypeParam：给具体参数哪个属性赋值，如Get方法需要给id赋值，参数层级比较深时，可以用冒号:隔开等。
> 4. transfors：递归给参数赋值，当参数层级比较深时，需要递归到参数所在的层级再进行赋值，比如datas.paramValues.id字段赋值，则需要递归到paramValues对象，给paramValues对象的id属性赋值 方法返回结果：`[{ruleId:"XX1",result:object1},{ruleId:"XX2",result:object2}]`

- Get、Fact方法： Get方法可以传递paramValues和sqlFields参数，paramValues只需定义主键信息，sqlFields可以定义查询的哪些列，Fact方法同Get完全一致，流程引擎执行Fact方法获取表单实体对象 例：

> 配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"Get","transfors":[{"paramName":"paramValues","transfors":[{"parentParamName":"paramValues","paramName":"id","transType":"eventData","transTypeParam":"id"}]}]}]}` 实际请求：`{"routeName":"SingleTemplateTest","applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"Get","ruleId":1,"objectName":"TemplateTests","paramModel":"local","datas":{"paramValues":{"id":"0b3cb8d7-6cb4-44bf-a480-2003bf4733b4"}}}]}` 说明：将触发事件绑定的对象的id字段赋值给datas.paramValues.id参数，执行默认的Get方法

- CreateOrUpdate、Create、Update方法 三个方法参数相同，一般情况调用CreateOrUpdate方法即可，新增或者修改数据库表数据，当id字段为空时，执行新增Create方法，当id字段不为空时，执行修改Update方法 例：

> 配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"CreateOrUpdate","transfors":[{"subFormId":"66e68bf9-8e05-4691-bf99-ed95c820260d","subViewId":"ecd0c03d-a67a-45a0-a64c-9c53e3522fb1","paramName":"paramValues","transType":""}]}]}}` 实际请求：`{"applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"CreateOrUpdate","ruleId":1,"objectName":"TemplateTests","paramModel":"local","datas":{"paramValues":{"id":"0b3cb8d7-6cb4-44bf-a480-2003bf4733b4","field1":"12","field2":"12",.....}}}]}` 说明：在ListView视图对话框保存按钮执行时，将子表单的子视图(ItemView)的数据传递到方法参数，执行CreateOrUpdate方法

- Delete方法 删除数据库一条数据 例：

> 配置：`{"ruleId":1,"objectName":"TemplateTests","methodId":"Delete","transfors":[{"paramName":"paramValues","transfors":[{"parentParamName":"paramValues","paramName":"id","transType":"eventData","transTypeParam":"id"}]}]}` 实际请求：`{"routeName":"SingleTemplateTest","applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"Delete","ruleId":1,"objectName":"TemplateTests","paramModel":"local","datas":{"paramValues":{"id":"499e1b14-5bd4-4d8d-bb72-46558996fbbd"}}}]}` 说明：在ListView的表格某一行点击删除按钮，执行删除方法，将点击事件绑定的对象的id字段赋值给参数datas.paramValues.id，执行删除方法

- DeleteWhere方法 按照自定义条件批量删除数据，比如ListView的批量选择删除功能，自定义条件参见附录【自定义查询条件】 例：

> 配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"DeleteWhere","transfors":[{"transType":"batchdelete"}]}]}` 实际请求：`{"applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"DeleteWhere","ruleId":1,"objectName":"TemplateTests","paramModel":"local","datas":{"sqlWheres":{"sqlExpressType":1,"children":[{"sqlExpressType":3,"field":"Id","conditionType":4,"value":["147632d6-7b12-411b-a7f0-c83744fb46c1","411c9856-07aa-4427-830c-35d6bfa29220"]}]}}}]}` 说明：transType为batchdelete，系统自动将用户批量勾选的数据id构造为数组赋值给参数，调用批量删除功能

- GetWhere方法 同Get方法，只是查询条件不是以id查询，是自定义的sqlWhere查询条件
- UpdateWhere方法 同Update方法，只是查询条件是自定义的sqlWhere查询条件

- PageList方法 ListView视图列表数据分页查询，将查询数据构造为查询条件参数，执行后端分页查询获取数据库数据，参见附录【自定义查询条件】 例：

> 配置：{"ruleId":1,"objectName":"TemplateTests","methodId":"PageList","userFields":"creatorId;lastModifierId;userField;userFields","remoteFieldInfos":[{"objectName":"OTN1","field":"singleField","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"OTN1","field":"singleFields","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"Student","field":"studentId","script":" ${obj.name}(${obj.userName}) "}],"transfors":[{"transType":"query"}]} 实际请求：{"routeName":"SingleTemplateTest","applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"PageList","ruleId":1,"objectName":"TemplateTests","userFields":"creatorId;lastModifierId;userField;userFields","remoteFieldInfos":[{"objectName":"OTN1","field":"singleField","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"OTN1","field":"singleFields","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"Student","field":"studentId","script":" ${obj.name}(${obj.userName}) "}],"paramModel":"local","datas":{"sqlSkipCount":0,"sqlMaxResultCount":15,"sqlWheres":{"sqlExpressType":1,"children":[{"sqlExpressType":3,"field":"Education","conditionType":5,"value":"master"},{"sqlExpressType":3,"field":"studentId","conditionType":1,"value":"05781136-4fef-4961-b5d7-d5b6073ddd37"},{"sqlExpressType":3,"field":"userFields","conditionType":5,"value":"39f9719b-0a41-6364-133c-e9c39c92d01a"}]}}}]} 说明：将ListView查询区域用户输入的值构造为参数参数，执行分页查询功能，userFields参数定义哪些字段是用户字段，需要显示用户名，remoteFieldInfos标识哪些字段是外键关联字段，需要关联显示外键信息，参见【外键关联处理】

- ListWhere方法 与PageList方法类似，只是去掉了分页功能，可以用在导出Excel等不需要分页获取数据集合的地方 例：

> 配置：{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"ListWhere","userFields":"creatorId;lastModifierId;userField;userFields","remoteFieldInfos":[{"objectName":"OTN1","field":"singleField","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"OTN1","field":"singleFields","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"Student","field":"studentId","script":" ${obj.name}(${obj.userName}) ","transfors":[{"transType":"exportexcel"}]}]} 实际请求：{"routeName":"SingleTemplateTest","applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"ListWhere","ruleId":1,"objectName":"TemplateTests","userFields":"creatorId;lastModifierId;userField;userFields","remoteFieldInfos":[{"objectName":"OTN1","field":"singleField","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"OTN1","field":"singleFields","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"Student","field":"studentId","script":" ${obj.name}(${obj.userName}) "}],"paramModel":"local","datas":{"sqlWheres":{"sqlExpressType":1,"children":[]},"excelDicts":[{"dict":"edu","field":"education"},{"dict":"sex","field":"sex"},{"dict":"post","field":"post"}],"excelName":"单表测试Excel数据","excelTemplate":[{"name":"string字段","field":"field1","fieldType":5,"isRequred":true,"validateType":0},{"name":"string2字段","field":"field2","fieldType":5,"validateType":0},{"name":"Int字段1","field":"fieldInt1","fieldType":1,"validateType":1},{"name":"int2字段","field":"fieldInt2","fieldType":1,"validateType":1},{"name":"datetime字段1","field":"fieldDatetime1","fieldType":7,"validateType":10},{"name":"datetime字段2","field":"fieldDatetime2","fieldType":7,"validateType":10},{"name":"bool字段","field":"fieldBool1","fieldType":4,"validateType":1},{"name":"学历","field":"education","fieldType":5,"validateType":11},{"name":"性别","field":"sex","fieldType":5,"validateType":11}]}}]} 说明：这个例子为导出Excel的配置，transType为exportexcel，excelName定义导出的Excel名称，excelDicts定义哪些字段是数据字典，excelTemplate定义导出Excel的模板信息，参见附录【导入导出Excel模版】其他参数同PageList方法说明

- MultiPageList方法 与PageList方法类似，PageList只能处理单表数据，对于多表关联处理无能为力（最多只显示关联表一个字段显示），此方法专门用于多表关联处理 例：

> 配置：{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"OTO1","methodId":"MultiPageList","transfors":[{"transType":"query","aliasInfos":"OTO1:a;OTO2:b","joinInfos":"OTO1 a LEFT JOIN OTO2 b ON a.Id=b.Id"}]}]} 实际请求：{"applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"MultiPageList","ruleId":1,"objectName":"OTO1","paramModel":"local","datas":{"sqlSkipCount":0,"sqlMaxResultCount":15,"sqlWheres":{"sqlExpressType":1,"children":[]},"joinInfos":"OTO1 a LEFT JOIN OTO2 b ON a.Id=b.Id","aliasInfos":"OTO1:a;OTO2:b"}}]} 说明：特殊使用参见【外键关联处理】，其他同PageList方法

- MultiListWhere 与ListWhere方法类似，ListWhere只能处理单表数据，对于多表关联处理无能为力（只多只显示关联表一个字段显示），此方法专门用于多表关联处理 说明：特殊使用参见【外键关联处理】，其他同ListWhere方法
- TreeListWhere 与ListWhere方法类似，此方法主要是针对树结构的数据库表进行查询，查询逻辑： 先根据查询条件查询出满足条件的数据，再把所有节点的祖先节点查询出来，构造为一树完整的树。 例：

> 配置：{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"Depts","methodId":"TreeListWhere","transfors":[{"transType":"querytree"}]}]} 实际请求：{"applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"TreeListWhere","ruleId":1,"objectName":"Depts","paramModel":"local","datas":{"sqlWheres":{"sqlExpressType":1,"children":[{"sqlExpressType":3,"field":"title","conditionType":5,"value":"后端"}]}}}]} 说明：获取树不会涉及到分页，整棵树完整数据显示，另外注意，树结构数据库对象自动添加维护Pld、TreeCode、Path、Icon、Title字段

- BatchCreate方法 批量新增数据，比如导入Excel，获取一对多关系，批量添加子表数据等场景。 例：

> 配置：{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"OTN1","methodId":"CreateOrUpdate","transfors":[{"subFormId":"e928438e-645f-45fb-9fe7-88c74dafaa00","subViewId":"8c78e079-c23d-4d27-907e-76eadfc30b05","paramName":"paramValues","transType":""}]},{"ruleId":2,"objectName":"OTN2","methodId":"BatchCreate","paramModel":"server","serverParams":[{"ruleId":"1","targetFields":"datas#oTN1_Id"}],"transfors":[{"subFormId":"e928438e-645f-45fb-9fe7-88c74dafaa00","subViewId":"852c8f20-c4f2-4e7f-bccb-1573d7a3af5c","transType":""}]}]} 说明：此配置执行一对多数据保存逻辑，执行两个方法，先执行主表的CreateOrUpdate方

> 法，再执行BatchCreate批量新增子表数据方法，BatchCreate的执行逻辑为：子表的数据是从子表单XX子视图XX获取（ListView表格数据），主表执行完CreateOrUpdate方法后，将结果（主键）值赋值给子表的oTN1_Id字段，再执行子表的BatchCreate批量新增子表数据方

- FormateTime(查询特定时间) 获取时间参数，返回查询开始时间和结束时间，具体参见获取特定查询时间，在报表、面板等处，执行后端方法时，赋值时间段参数 例： 请求： `{"routeName":"M-Simple_t","applicationCode":"Default","methods":[{"methodId":"FormateTime","objectName":"BaseObject","ruleId":1,"datas":{"timeType":1}}]}` 请求结果：`[{"result":{"startTime":"2022-07-28 00:00:00","endTime":"2022-07-28 23:59:59"},"ruleId":1}]`
- CurrentUserInfo(获取当前登录用户信息) 获取当前登录的用户信息 请求： `{"routeName":"M-Simple_t","applicationCode":"Default","methods":[{"methodId":"CurrentUserInfo","objectName":"BaseObject","ruleId":1,"datas":{}}]}` 请求结果： `[{"result":{"UserId":"503e4458-8998-16e9-8243-39f586ee75ce","UserName":"admin","DeptId":"1","Name":"管理员","Roles":["StockLook","CompanyManager","FManager","MiddleManager","staff","FDeptManater","HR","DeptManater","Leader","admin"]},"ruleId":1}]`
- CurrentUserDeptInfo(获取登录用户部门信息) 获取登录用户的部门信息 请求： `{"routeName":"M-Simple_t","applicationCode":"Default","methods":[{"methodId":"CurrentUserDeptInfo","objectName":"BaseObject","ruleId":1,"datas":{}}]}` 请求答复： `[{"result":{"Id":"1","Pld":"0","Name":"成都东宸科技有限公司","Path":"成都东宸科技有限公司","TreeCode":"0."},"ruleId":1}]`
- UserInfo(获取特定用户信息) 获取特定用户信息 请求： `{"routeName":"M-Simple_t","applicationCode":"Default","methods":[{"methodId":"CurrentUserInfo","objectName":"BaseObject","ruleId":1,"datas":{"userId":"503e4458-8998-16e9-8243-39f586ee75ce"}}]}` 请求结果： `[{"result":{"UserId":"503e4458-8998-16e9-8243-39f586ee75ce","UserName":"admin","DeptId":"1","Name":"管理员","Roles":["StockLook","CompanyManager","FManager","MiddleManager","staff","FDeptManater","HR","DeptManater","Leader","admin"]},"ruleId":1}]`
- UserDeptInfo(获取特定部门信息) 获取特定部门信息 请求： `{"routeName":"M-Simple_t","applicationCode":"Default","methods":[{"methodId":"CurrentUserDeptInfo","objectName":"BaseObject","ruleId":1,"datas":{"deptId":"1"}}]}` 请求答复： `[{"result":{"Id":"1","Pld":"0","Name":"成都东宸科技有限公司","Path":"成都东宸科技有限公司","TreeCode":"0."},"ruleId":1}]`

3. 绑定数据(actionType=3) 绑定数据主要是定义数据从一个地方绑定到另一个地方，一个绑定数据规则可以定义执行多条绑定逻辑 绑定数据由两个参数：fromTrans:定义绑定数据来源，fromTransType：值为1,2,3,4,5，1代表绑定数据源为方法执行结果，2代表绑定数据源从其他控件获取，3代表绑定数据源从触发事件绑定的数据获取,4代表从表达式获取值,5代表从当前登录用户获取，toTrans:定义将数据绑定到哪里

例1：

> 配置： `[{"fromTrans":{"methodRuleId":1,"fromTransType":1},"toTrans":{"subFormId":"084beddb-c518-46df-8647-de73739c3631","subViewId":"d2b0b615-4678-462d-a795-34bb3b46f75e"}}]` 说明：此配置用在ListView列表视图，点击列表编辑按钮触发的事件的其中一个规则执行，前面还有设置弹出框显示规则执行和执行后端get方法，此规则定义从get方法获取结果赋值给子表单XX子视图XX，即将Get方法结果赋值给编辑表单的ItemView编辑视图，ItemView编辑视图即可绑定表单数据。

例2：

> 配置： `[{"fromTrans":{"propertyName":"id","fromTransType":3},"toTrans":{"subFormId":"af1756bb-da48-4b5f-821d-78aeeaddbd2e","subViewId":"b022e1c3-ea6f-4f59-b2ef-67efec2701e0","propertyName":"oTN1_Id","type":"setparam"}}]` 说明：此配置用于一对多对主表数据进行编辑，将编辑主表时主表Id传递给子表的列表表单ListView视图的参数，参数名为oTN1_Id

例3：

> 配置： `[{"fromTrans":{"fromTransType":2,"transType":"getparam","propertyName":"oTN1_Id"},"toTrans":{"subFormId":"084beddb-c518-46df-8647-de73739c3631","subViewId":"d2b0b615-4678-462d-a795-34bb3b46f75e","propertyName":"oTN1_Id"}}]` 说明：此配置用于一对多对主表数据进行编辑，编辑时可以新增子表数据，新增或者编辑子表数据规则自动将主表的Id赋值给子表的oTN1_Id，大致逻辑： 打开主表编辑对话框，将主表Id赋值给子表表单的ListView视图的参数，新增子表时，将ListView参数oTN1_Id的值赋值给子表编辑表单的ItemView视图的oTN1_Id字段 。

例4：

> 配置： `[{"fromTrans":{"fromTransType":3},"toTrans":{"subViewId":"e79df39e-d855-4a86-880d-046f60a9e2e8","propertyName":"reset"}}]` 说明：弹出新增数据对话框，重置子视图ItemView的值

例5：

> 配置： `[{"fromTrans":{"fromTransType":4,"expressionType":0,"config":"新增"},"toTrans":{"subFormId":"bbb537c4-078e-4694-882a-b7677057b198","subViewId":"cea92081-7f3b-420d-a5c2-48912a4435ec","propertyName":"state"}}]` 说明：例子功能为将子表单的编辑子视图的状态字段赋值为新增

4. 停止执行(actionType=5) 停止配置为空，当执行到停止执行Action时，结束事件后续规则执行，直接中断事件
5. 执行其他规则(actionType=7) 触发其他事件，执行到此规则时，先触发新的事件，执行新的事件的一系列规则，再接着执行后续本事件的规则 例：

> 配置： `{"eventName":"click"}` 对象：查询按钮Id 说明：此配置用于ListView列表视图，当执行到此规则时，触发查询按钮的click事件，先执行查询按钮的click事件配置的规则，再执行后续规则

6. 弹出消息提示(actionType=14) 弹出消息提示，提示错误、警告、提示等消息，参见ant消息提示 例：

> 配置： `{"content":"数据保存成功","msgType":"success"}` 说明：此配置用于ItemView编辑视图保存方法执行成功好提示用户

7. 条件判断(actionType=4) 定义一个表达式，和一个trueActions集合和falseActions集合，表达式执行结果为true时，执行trueActions的规则集合，表达式结果为false时，执行falseActions的规则集合，执行完成之后，再接着执行条件判断规则后续规则。 表达式定义如下：

- expressionType：定义表达式类型
- config：定义表达式配置
- expressionType为同值具体说明：

> 值为0：固定值，定义一个常量，比如true,1,'aa'等 值为1：值从方法执行结果获取，config配置：methodRuleId定义方法ruleId，propertyName定义从方法结果哪个属性获取值，多层级用冒号:隔开。 值为2：从对象获取值，可以从不同的表单、视图、控件获取特定的值，不同的对象定义的可获取值不同。config配置对象信息以及额外的参数 值为3：从事件绑定的数据获取，config的propertyName定义从事件对象的（触发事件时，会将特定事件的关联数据绑定到事件对象中，比如点击列表视图某行的编辑按钮，会将当前行的数据绑定的事件对象中）哪个属性获取值，多层级用冒号:隔开 值为4：定义转换函数，将数据从一种类型转换为另外一种类型。config的配置：expressionFuncName，转换的目标类型，bool、string、date、number，后续还可以扩展其他特定转换函数，children，子表达式集合，转换函数只有一个子表达式 值为5：定义关系表达式，执行结果为true或者false，config的配置：expressionFuncName，关系运算符，=、!=、>、>=、<、<=。children，子表达式集合，转换函数只有两个子表达式，第一个子表达式定义关系运算左边的值，第二个表达式结果定义关系表达式右边的值。 值为6：定义逻辑表达式，执行结果为true或者false，config的配置：expressionFuncName，逻辑运算符&&、||、！，&&和||可定义多可children，多个children的结果执行&&或者||运算，!只能定义一个children 值为7：集合相关函数计算，config的配置：expressionFuncName，length获取集合数量，后续可定义其他函数，children子表达式集合只有一个

8. 弹出确认提示(actionType=6) 同条件判断规则，只是这里弹出用户确认对话框，提示用户选择是否确认，再定义trueActions和falseActions规则集合 例：

> 配置：`{"content":"是否确认删除？","trueActions":[{"actionType":2,"actionConfig":{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"Depts","methodId":"Delete","transfors":[{"paramName":"paramValues","transfors":[{"parentParamName":"paramValues","paramName":"id","transType":"eventData","transTypeParam":"id"}]}]}]},{"actionType":7,"objId":"fd616db9-52f1-4f40-84f6-6a5c6016753d","actionConfig":{"eventName":"click"}}],"falseActions":[{"actionType":5}]}` 说明：弹出是否确认删除提示，用户选择是，则执行删除方法，然后触发查询按钮点击事件，执行查询按钮点击事件规则，用户选择否，则停止执行

9. 重新加载页面(actionType=8) 重新加载页面会销毁前端所有组件，重新渲染界面，即重新载入vue页面，相当于第一次打开某个页面。常常绑定到列表页面的“刷新”按钮上。
10. 执行导出Excel方法(actionType=9) 同【执行方法(actionType=2)】方法参数，导出Excel通常执行ListWhere后端默认方法，另外会额外传递Excel模版信息参数，模版配置定义在ListView的属性设置中,exportexcel。 例：

> 配置：{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"ListWhere","transfors":[{"transType":"exportexcel"}]}]} 实际请求：{"routeName":"SingleTemplateTest","applicationCode":"Default","isTransaction":true,"methods":[{"methodId":"ListWhere","ruleId":1,"objectName":"TemplateTests","paramModel":"local","datas":{"sqlWheres":{"sqlExpressType":1,"children":[]},"excelDicts":[{"dict":"edu","field":"education"},{"dict":"sex","field":"sex"},{"dict":"post","field":"post"}],"excelName":"单表测试Excel数据","excelTemplate":[{"name":"string字段","field":"field1","fieldType":5,"isRequred":true,"validateType":0},{"name":"string2字段","field":"field2","fieldType":5,"validateType":0},{"name":"Int字段1","field":"fieldInt1","fieldType":1,"validateType":1},{"name":"int2字段","field":"fieldInt2","fieldType":1,"validateType":1},{"name":"datetime字段1","field":"fieldDatetime1","fieldType":7,"validateType":10},{"name":"datetime字段2","field":"fieldDatetime2","fieldType":7,"validateType":10},{"name":"bool字段","field":"fieldBool1","fieldType":4,"validateType":1},{"name":"学历","field":"education","fieldType":5,"validateType":11},{"name":"性别","field":"sex","fieldType":5,"validateType":11}]}]}

11. 执行导出Excel模版方法(actionType=13) 同【执行导出Excel方法(actionType=9)】方法，只是这里的transType值为downexceltemplate 例：

> 配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"ListWhere","transfors":[{"transType":"downexceltemplate"}]}]}`

12. 获取数据(actionType=12) 一般情况用在条件表达式规则执行的地方，另外可以用于流程引擎执行表单验证方法（不是需要获取值，只是触发表单验证） 例：

> 配置：`{"fromTransType":2,"fromTrans":{"transType":"validate"}}`

13. 打开工作流(actionType=10) 从表单处，有两种方式打开工作流，一种是发起流程，一种是查询流程，打开流程管理对话框 例1：

> 配置：`{"isCreate":true,"flowDefineId":"5da7e6b8d6b444c9866b15ab028dd6c6"}` 说明：发起流程，isCreate标识为创建流程，flowDefineId为流程定义Id，发起时，找流程定义最新发布的流程版本发起流程实例 例2： 配置：`{"formIdFieldName":"id"}` 说明：以表单Id打开流程管理界面，流程都会挂载一个表单，传递表单Id可以找到流程实例，然后打开流程管理界面。

14. 获取后端方法执行(actionType=11) 此规则同【执行方法(actionType=2)】，这里只是获取执行方法的参数，不正真的触发后端方法执行，真正的执行在其他地方控制执行。目前只用在流程引擎提交或者保存时，流程引擎需要同时保存表单方法和流程数据，在流程引擎执行时调用自定义表单方法。 例：

> 配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"Vacation","methodId":"CreateOrUpdate","paramModel":"local","execType":"workflow_formId","transfors":[{"paramName":"paramValues","transType":""}]},`

{"ruleId":2,"objectName":"Vacation","methodId":"Fact","paramModel":"server","execType":"workflow_fact","serverParams":[{"ruleId":"1","targetFields":"datas:paramValues:id"}]}]}

06-特殊应用场景说明

很多复杂、高级、特殊的一些场景应用都在此文章中找得到，这篇文章经常会更新。

外键关联处理

1. 外键关联的地方，数据库一般存储的是外键的Id，但是在编辑视图的列、列表视图的查询、列表视图的表格列显示等地方，需要自定义格式化显示关联的其他表的信息，这个时候就需要特殊处理（数据字典数据库存储的是字典项的Code，显示的是字典项的值，这部分由自定义表单自动处理，不需要另外干涉）
2. 在编辑视图的列、列表视图的查询这些地方，自定义表单的控件使用已经做了封装，参见附录【自定义表单控件使用说明】说明
3. 列表视图的表格列显示，需要在执行规则的时候，调用后端方法时，指定哪些此属于外键关联，并关联到哪些列，这样，返回的数据时，系统将会另外增加“xxx_Name”或者“xxx_UserName”字段，列表字段绑定的时候，绑定额外增加的字段 例：

> 方法执行配置：`{"aplicationCode":"Default","isTransaction":true,"methods":[{"ruleId":1,"objectName":"TemplateTests","methodId":"PageList","userFields":"creatorId;lastModifierId;userField;userFields","remoteFieldInfos":[{"objectName":"OTN1","field":"singleField","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"OTN1","field":"singleFields","script":" ${obj.string1}(${obj.sex}) "},{"objectName":"Student","field":"studentId","script":" ${obj.name}(${obj.userName}) "}],"transfors":[{"transType":"query"}]}]}` 列表绑定配置：`"columns":[...{"align":"center","dataIndex":"studentId_Name","title":"关联学生"},{"align":"center","dataIndex":"userFields_UserName","title":"多选用户"},{"align":"center","dataIndex":"userField_UserName","title":"单选用户"},{"align":"center","dataIndex":"creatorId_UserName","title":"创建者"}]` 说明：userFields为存储用户的列，remoteFieldInfos为外键关联的关联信息

Wrap对象定义

Wrap对象往往是对子视图或者子表单的外层样式封装，可以是对话框、Div等，每个Wrap对象会定义一个Id，Wrap对象在规则链中的Id为“Wrap所在的视图或者表单_Wrap的Id”，设置规则时需要特别注意

多表关联处理

多表管理实际为拼接Sql语句，默认方法MultiPageList和MultiListWhere用于专门处理多表关联查询应用场景，参数aliasInfos用于定义对象与别名之间的映射，joinInfos定义关联查询语句，所有在使用字段的地方，字段都变为了"原始字段_别名"，使用时，需要特别注意，在使用查询条件、界面绑定字段值时或者导入导出Excel时，这个时候都不是使用原始的Object对象字段名称，而是加上别名后的名称，如string1_a,education_a,int1_b等，注意这里做了Sql注入判断。 例：

```
{"rowSelection":{},"rowKey":"id_a","columns":[{"align":"center","dataIndex":"string1_a","title":"字符1","width":"4"},{"align":"center","dataIndex"
```

流程引擎与自定义表单关联

1. 每一个流程会有一个流程定义Id，一个流程可能会有多个版本，但只能有一个发布版本，同一个流程多个版本流程定义Id相同。
2. 每一个流程都会挂接一个表单，新建流程时，会填写关联的自定义表单Id，流程实例会存储表单对应的对象主键；对象实例会自动增加流程实例Id字段，存储流程实例Id，这样，流程实例与自定义表单对象都相互存储了对方的Id，流程引擎与自定义表单就建立了关联关系。
3. 发起流程时，执行打开工作流规则，传递流程定义Id；自定义表单打开工作流时，传递对象主键即可找到流程实例信息，进而管理流程实例。
4. 流程管理界面打开流程时，通过表单定义Id渲染自定义表单，用表单Id查找自定义表单实例，绑定界面数据。

自动编号

自动编号可以支持雪花算法和流水号算法，默认为雪花算法

1. 雪花算法： 每一个对象字段维护自己的雪花算法器，在对象字段类型选择AutoNo，系统在新增数据时，后台自动生成No，在编辑对象字段属性时，可以填写默认值，做为自动生成的No的前置字符串
2. 流水号算法： 每一个对象每一个字段维护自己的流水算法，在对象字段类型选择AutoNo，填写“字段配置”设置，如:{"serialNumber":{"preNo":"ST","length":6,"useDay":true,"useMonth":false,"useYear":false}}，生成字段如ST20220617000006。【说明：serialNumber标识采用流水算法，preNo为前缀，length标识流水长度，useDay标识按照每天生成，useMonth标识按照每月生成，useYear标识按年生成】 模版生成的表单，自动将自动生成的编辑控件设置为不可用，其他情况，当作字段类型为String使用

操作权限

1. 在模块管理处增加操作权限，设置操作权限编码，操作权限编码在系统内唯一，定义操作关联的角色。
2. 在控件绑定的地方设置控件属性v-action:"操作权限编码"，如："v-action": "TemplateTests_Add"，即可控制操作权限（v-action或者action都可以）

远程控件参数固定

远程控件参数会根据用户输入的值替换查询参数的值，如果查询参数是固定的不随用户输入值改变，增加isFixed属性控制（下拉搜索控件一般会定定义好查询的条件，动态替换查询的字段值，但有些情况查询字段的值需要固定）。 例：

{"placeholder":"选择库区","allowClear":true,"componentName":"a-select","sourceSettings":{"isSpriteMethod":true,"params":{"isTransaction":true,

表单冗余数据

下拉选择其他表数据，将特定的字段绑定到本编辑视图其他字段等场景使用，大致步骤：1.设置冗余字段控件不可编辑.2.检测控件chenge事件3.新增规则，将下拉选择控件的值绑定到编辑视图特定字段值，参见 体验网站 > 表单常规示例 > 控件综合应用 例：

> 配置：`[{"fromTrans":{"fromTransType":2,"transType":"selected","propertyName":"itemCode","objId":"52ccc411-b16b-4974-8e02-bb87816244bc"},"toTrans":{"propertyName":"itemCode"}}]` 说明：注意fromTrans的transType为selected，表示取下拉控件目前选择的对象信息，例子的意思是将下拉选择对象的itemCode值取出来绑定到编辑视图的itemCode字段中

菜单参数

设置页面参数，进入一个页面，可能会形成一棵庞大的对象树，有时传递数据会非常麻烦，这时可以定义页面级的参数，所有对象共享本页面菜单上定义的自定义参数 例：

> 在1对多Tab表单应用场景，打开一个编辑页面，需要将主表的Id传递给每个Tab包裹的表单或者视图里面，这里可以在主表列表页面编辑按钮事件，将点击行的Id传递到菜单参数，各个Tab页面各个规则使用时，直接从菜单获取传递的值 编辑按钮配置：`[{"fromTrans":{"propertyName":"id","fromTransType":3},"toTrans":{"isMenu":true,"propertyName":"oTNO_ts_Id"}}]` 子表单查询参数设置：`[{"fromTrans":{"propertyName":"oTNO_ts_Id","isMenu":true},"toTrans":{"propertyName":"oTNO_ts_Id"}}]` 说明：isMenu表示设置或者获取菜单参数值

自定义绑定事件参数

对于特定的一些事件，系统在触发事件的时候，会自动设置相应的参数做为事件参数，在事件的任何一个规则执行时，可以获取事件的参数，比如下拉控件change事件（事件参数为下拉值），列表编辑、删除按钮点击事件（事件参数为行数据），在绑定数据规则执行时，可以将其他源的数据绑定到事件参数中，在后续规则执行时，可以直接获取绑定的值。 例：

> `[{"fromTrans":{"methodRuleId": 1,"fromTransType": 1},"toTrans":{"propertyName":"id","isEvent": true}}]` 说明：isEvent标识是绑定值到事件参数，此例子用在新增主表数据完成之后，直接打开编辑视图对话框，由于编辑对话框事件需要设置事件参数包含id字段，这里直接将新增方法保存之后的值绑定的事件参数的id字段。

删除子表数据

删除主表数据时，删除子表数据场景 例：

> 配置：{"content":"是否确认删除？","trueActions":[{"actionType":2,"actionConfig":{"methods":[{"ruleId":1,"objectName":"OTNO_ts","methodId":"Delete","transfors":[{"paramName":"paramValues","transfors":[{"parentParamName":"paramValues","paramName":"id","transType":"eventData","transTypeParam":"id"}]}]},{"methodId":"DeleteWhere","ruleId":2,"objectName":"OTNN1_ts","paramModel":"local","transfors":[{"paramName":"sqlWheres","transfors":[{"parentParamName":"sqlWheres","paramName":"children:0","transfors":[{"parentParamName":"children:0","paramName":"value","transType":"eventData","transTypeParam":"id"}]}]}],"baseConfig":{"sqlWheres":{"sqlExpressType":1,"children":[{"sqlExpressType":3,"field":"oTNO_ts_Id","conditionType":1,"value":null}]}}}]},{"actionType":7,"actionConfig":{"eventName":"refresh"}}],"falseActions":[{"actionType":5}]} 说明：先配置基础删除子表数据的配置，绑定方法参数时，根据删除的主表id值替换删除子表外键字段的值，达到删除子表数据作用。参见附录【自定义查询条件】

07-表单模板

常规的业务，在需求以及数据库设计完成之后，可能就仅仅在界面上几分钟的配置就能够完成所有的开发、测试、部署工作，完全解放繁琐的CRUD工作 表单模板能够快速创建常规的业务模块，系统尽量将常规的业务功能做成模板，方便快速的创建业务模块功能，选择一个模板之后，会将模板对应的表 单、子表单、子视图、控件等所有自定义表单相关的定义全部自动创建出来。

设计参考

表单模版相关设计

- 模板表单选择 选择定义为模板的表单，列表显示为“表单名称(表单备注)”，通常为进入业务的根表单，模板表单备注规范：表单类型中文->对象Map(,)，描述Map(,**)，如：单表_模板列表表单->对象Map(Simple_ts)，描述Map(单表_模板)
- 对象Map 需要替换为对象名称
- 描述Map 需要替换的显示名称
- 列表排除字段 有些特殊字段在对象属性定义了，不需要默认在ListView和ItemView里面自动创建出来时使用
- 业务分类 自己定义业务的分类，模板创建的表单和视图的业务分类都为此值
- Item表单行 格式为：New对象名1:编辑视图列数;New对象名2:编辑视图列数，列数只支持1,2,3

具体的内置模板请参考【体验网站】

99-附录

表单验证

编辑视图字段验证参考ant表单校验规则 自定义表单内置的正则表达式验证定义如下：

- 正则表达式定义：

```
export const RegRule = { // 正则存储
  Email: /^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$/,
  Tel: /^((\(\d{3,4}\)|\d{3,4}-|\s)?\d{7,14}$/,
  Phone: /^1(3|4|5|6|7|8|9)\d{9}$/,
  IDCard: /^[1-9]\d{5}(18|19|20|(3\d))\d{2}((0[1-9])|(1[0-2]))(([0-2][1-9])|10|20|30|31)\d{3}[0-9Xx]$/,
  Num100: /^(?:0|[1-9][0-9]?|100)$/, // 0-100的整数
  CommonStr_: /^[A-Za-z0-9_]+$/, // 字母，数组，下划线组合
  CommonStr: /^[A-Za-z0-9]+$/, // 字母，数组
  CarNumber: /^[京津沪渝冀豫云辽黑湘皖鲁新苏浙赣鄂桂甘晋蒙陕吉闽贵粤青藏川宁琼使领A-Z]{1}[A-Z]{1}[A-Z0-9]{4}[A-Z0-9挂学警港澳]{1}$/, // 车牌号
  QQ: /^[1-9][0-9]{4,10}$/, // QQ号码
  WeiChart: /^[a-zA-Z]([-_a-zA-Z0-9]{5,19})+$/, // 微信号
  IPV4: /^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?).){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/, // IP地址
  Decimal61: /^-?(0|[1-9][0-9]{0,5})(.[0-9]{1,1})?$/,
  Decimal62: /^-?(0|[1-9][0-9]{0,5})(.[0-9]{1,2})?$/,
  Decimal63: /^-?(0|[1-9][0-9]{0,5})(.[0-9]{1,3})?$/,
  Decimal64: /^-?(0|[1-9][0-9]{0,5})(.[0-9]{1,4})?$/,
  Decimal66: /^-?(0|[1-9][0-9]{0,5})(.[0-9]{1,6})?$/,
  Decimal81: /^-?(0|[1-9][0-9]{0,7})(.[0-9]{1,1})?$/,
  Decimal82: /^-?(0|[1-9][0-9]{0,7})(.[0-9]{1,2})?$/,
  Decimal83: /^-?(0|[1-9][0-9]{0,7})(.[0-9]{1,3})?$/,
  Decimal84: /^-?(0|[1-9][0-9]{0,7})(.[0-9]{1,4})?$/,
  Decimal86: /^-?(0|[1-9][0-9]{0,7})(.[0-9]{1,6})?$/,
  Decimal101: /^-?(0|[1-9][0-9]{0,9})(.[0-9]{1,1})?$/,
  Decimal102: /^-?(0|[1-9][0-9]{0,9})(.[0-9]{1,2})?$/,
  Decimal103: /^-?(0|[1-9][0-9]{0,9})(.[0-9]{1,3})?$/,
  Decimal104: /^-?(0|[1-9][0-9]{0,9})(.[0-9]{1,4})?$/,
  Decimal106: /^-?(0|[1-9][0-9]{0,9})(.[0-9]{1,6})?$/,
  Decimal121: /^-?(0|[1-9][0-9]{0,11})(.[0-9]{1,1})?$/,
  Decimal122: /^-?(0|[1-9][0-9]{0,11})(.[0-9]{1,2})?$/,
  Decimal123: /^-?(0|[1-9][0-9]{0,11})(.[0-9]{1,3})?$/,
  Decimal124: /^-?(0|[1-9][0-9]{0,11})(.[0-9]{1,4})?$/,
  Decimal126: /^-?(0|[1-9][0-9]{0,11})(.[0-9]{1,6})?$/,
  Double: /^-?(0|[1-9][0-9]*)(.[0-9]*)?$/, // Double
  PDecimal61: /^(0|[1-9][0-9]{0,5})(.[0-9]{1,1})?$/,
  PDecimal62: /^(0|[1-9][0-9]{0,5})(.[0-9]{1,2})?$/,
  PDecimal63: /^(0|[1-9][0-9]{0,5})(.[0-9]{1,3})?$/,
  PDecimal64: /^(0|[1-9][0-9]{0,5})(.[0-9]{1,4})?$/,
  PDecimal66: /^(0|[1-9][0-9]{0,5})(.[0-9]{1,6})?$/,
  PDecimal81: /^(0|[1-9][0-9]{0,7})(.[0-9]{1,1})?$/,
  PDecimal82: /^(0|[1-9][0-9]{0,7})(.[0-9]{1,2})?$/,
  PDecimal83: /^(0|[1-9][0-9]{0,7})(.[0-9]{1,3})?$/,
  PDecimal84: /^(0|[1-9][0-9]{0,7})(.[0-9]{1,4})?$/,
  PDecimal86: /^(0|[1-9][0-9]{0,7})(.[0-9]{1,6})?$/,
  PDecimal101: /^(0|[1-9][0-9]{0,9})(.[0-9]{1,1})?$/,
  PDecimal102: /^(0|[1-9][0-9]{0,9})(.[0-9]{1,2})?$/,
  PDecimal103: /^(0|[1-9][0-9]{0,9})(.[0-9]{1,3})?$/,
  PDecimal104: /^(0|[1-9][0-9]{0,9})(.[0-9]{1,4})?$/,
  PDecimal106: /^(0|[1-9][0-9]{0,9})(.[0-9]{1,6})?$/,
  PDecimal121: /^(0|[1-9][0-9]{0,11})(.[0-9]{1,1})?$/,
  PDecimal122: /^(0|[1-9][0-9]{0,11})(.[0-9]{1,2})?$/,
  PDecimal123: /^(0|[1-9][0-9]{0,11})(.[0-9]{1,3})?$/,
  PDecimal124: /^(0|[1-9][0-9]{0,11})(.[0-9]{1,4})?$/,
  PDecimal126: /^(0|[1-9][0-9]{0,11})(.[0-9]{1,6})?$/,
  PDouble: /^(0|[1-9][0-9]*)(.[0-9]*)?$/,
  Integer1: /^-?(0|[1-9])$/,
  Integer2: /^-?(0|[1-9][0-9]{0,1})$/,
  Integer4: /^-?(0|[1-9][0-9]{0,3})$/,
  Integer8: /^-?(0|[1-9][0-9]{0,7})$/,
  Integer12: /^-?(0|[1-9][0-9]{0,11})$/,
  Integer16: /^-?(0|[1-9][0-9]{0,15})$/,
  Integer20: /^-?(0|[1-9][0-9]{0,19})$/,
```

```
  PInteger1: /^(0|[1-9])$/,
  PInteger2: /^(0|[1-9][0-9]{0,1})$/,
  PInteger4: /^(0|[1-9][0-9]{0,3})$/,
  PInteger8: /^(0|[1-9][0-9]{0,7})$/,
  PInteger12: /^(0|[1-9][0-9]{0,11})$/,
  PInteger16: /^(0|[1-9][0-9]{0,15})$/,
  PInteger20: /^(0|[1-9][0-9]{0,19})$/,
}
```

- 正则表达式错误提示：

```
export const RegRuleMsg = {
  Email: '邮箱地址填写错误',
  Tel: '电话号码填写错误',
  Phone: '手机号码填写错误',
  IDCard: '身份证号码填写错误',
  Num100: '请输入0-100之间的数字',
  CommonStr_: '只能输入字母、数字、下划线组成的字符',
  CommonStr: '只能输入字母、数字组成的字符',
  CarNumber: '车牌号输入错误',
  QQ: 'QQ号码输入错误',
  WeiChart: '微信号输入错误',
  IPV4: 'IP地址输入错误',
  Decimal61: '请输入最多6位整数，最多1位小数组成的浮点数',
  Decimal62: '请输入最多6位整数，最多2位小数组成的浮点数',
  Decimal63: '请输入最多6位整数，最多3位小数组成的浮点数',
  Decimal64: '请输入最多6位整数，最多4位小数组成的浮点数',
  Decimal66: '请输入最多6位整数，最多6位小数组成的浮点数',
  Decimal81: '请输入最多8位整数，最多1位小数组成的浮点数',
  Decimal82: '请输入最多8位整数，最多2位小数组成的浮点数',
  Decimal83: '请输入最多8位整数，最多3位小数组成的浮点数',
  Decimal84: '请输入最多8位整数，最多4位小数组成的浮点数',
  Decimal86: '请输入最多8位整数，最多6位小数组成的浮点数',
  Decimal101: '请输入最多10位整数，最多1位小数组成的浮点数',
  Decimal102: '请输入最多10位整数，最多2位小数组成的浮点数',
  Decimal103: '请输入最多10位整数，最多3位小数组成的浮点数',
  Decimal104: '请输入最多10位整数，最多4位小数组成的浮点数',
  Decimal106: '请输入最多10位整数，最多6位小数组成的浮点数',
  Decimal121: '请输入最多12位整数，最多1位小数组成的浮点数',
  Decimal122: '请输入最多12位整数，最多2位小数组成的浮点数',
  Decimal123: '请输入最多12位整数，最多3位小数组成的浮点数',
  Decimal124: '请输入最多12位整数，最多4位小数组成的浮点数',
  Decimal126: '请输入最多12位整数，最多6位小数组成的浮点数',
  Double: '请输入浮点数字',
  PDecimal61: '请输入最多6位整数，最多1位小数组成的正浮点数',
  PDecimal62: '请输入最多6位整数，最多2位小数组成的正浮点数',
  PDecimal63: '请输入最多6位整数，最多3位小数组成的正浮点数',
  PDecimal64: '请输入最多6位整数，最多4位小数组成的正浮点数',
  PDecimal66: '请输入最多6位整数，最多6位小数组成的正浮点数',
  PDecimal81: '请输入最多8位整数，最多1位小数组成的正浮点数',
  PDecimal82: '请输入最多8位整数，最多2位小数组成的正浮点数',
  PDecimal83: '请输入最多8位整数，最多3位小数组成的正浮点数',
  PDecimal84: '请输入最多8位整数，最多4位小数组成的正浮点数',
  PDecimal86: '请输入最多8位整数，最多6位小数组成的正浮点数',
  PDecimal101: '请输入最多10位整数，最多1位小数组成的正浮点数',
  PDecimal102: '请输入最多10位整数，最多2位小数组成的正浮点数',
  PDecimal103: '请输入最多10位整数，最多3位小数组成的正浮点数',
  PDecimal104: '请输入最多10位整数，最多4位小数组成的正浮点数',
  PDecimal106: '请输入最多10位整数，最多6位小数组成的正浮点数',
  PDecimal121: '请输入最多12位整数，最多1位小数组成的正浮点数',
  PDecimal122: '请输入最多12位整数，最多2位小数组成的正浮点数',
  PDecimal123: '请输入最多12位整数，最多3位小数组成的正浮点数',
  PDecimal124: '请输入最多12位整数，最多4位小数组成的正浮点数',
  PDecimal126: '请输入最多12位整数，最多6位小数组成的正浮点数',
  PDouble: '请输入正浮点数字',
  Integer1: '请输入最多1位整数',
  Integer2: '请输入最多2位整数',
  Integer4: '请输入最多4位整数',
  Integer8: '请输入最多8位整数',
```

```
  Integer12: '请输入最多12位整数',
  Integer16: '请输入最多16位整数',
  Integer20: '请输入最多20位整数',
  PInteger1: '请输入最多1位正整数',
  PInteger2: '请输入最多2位正整数',
  PInteger4: '请输入最多4位正整数',
  PInteger8: '请输入最多8位正整数',
  PInteger12: '请输入最多12位正整数',
  PInteger16: '请输入最多16位正整数',
  PInteger20: '请输入最多20位正整数',
}
```

自定义表单控件使用说明*

1. 自定义表单控件使用参考示例 > 体验网站> 控件综合应用
2. 自定义表单控件至关重要，丰富的控件是其他业务的基石。 *控件整体的设计实现思路：对ant的控件进行二次封装，统一包装为自定义表单能识别的控件对象，能够在规则引擎中直接引用，且控件触发的事件自动抛出自定义表单的事件，如果自定义表单定义了对应的规则触发事件，则标识命中控件的事件，进而执行规则配置；规则配置的执行目标也可以选择自定义的控件做为执行对象；比如新增按钮点击事件、视图加载后设置订单状态字段为禁用且设置订单状态值为初始等规则*
3. 控件在使用的时候，都是先定义控件名称，再定义控件的属性设置。

- 控件名称：col-form-common 主要是对ant控件的包装，设置componentName字段为具体使用的哪个控件，如a-input，a-date-picker等，控件属性设置如： `{'placeholder':'请输入电话的数据','componentName':'a-input'}` ，且可以添加一些自定义的属性，定义特定控件的使用
- 控件名称：col-form-dict（可以由col-form-common控件代替） 字典选择控件，内置使用ant的a-select控件，控件设置时，需要设置dict属性，控件属性设置如： `{'placeholder':'请选择性别信息', 'dict':'sex','componentName':'a-select'}`
- 控件名称：col-form-select（带搜索框的下拉选择控件） 带搜索框的下拉选择控件，支持搜索后单选或者多选择。
- 控件名称：col-form-tree-select（ *弃用* ） 直接使用col-form-common控件即可，下拉树选择，内置使用a-tree-select控件，需要通过规则填充下拉控件值，控件属性设置如： `{"placeholder":"请选择部门","componentName":"a-tree-select"}`
- 控件名称：col-form-user 系统用户搜索选择控件，支持文本搜索和文本+部门高级选择搜索，支持多选和单选
- 典型业务的配置说明

写在前面，特殊配置使用说明

searchEventName，这个字段主要用在ListView的查询区域，如果配置了，则代表用户操作某个控件，触发了指定的事件（事件名称同ant-design相同）时，自动触发列表查询。 dict，如果是数据字典，则设置属性值，值为字典编码，系统自动查找数据字典项绑定到控件或者在需要的地方显示字典显示值（数据库存储的是字典项编码） sourceSettings，这个属性为一个对象，如果需要远程调用的地方，都需要设置此对象值

> dict：如果是数据字典，设置此值，值为字典编码，此处设置为调用后端字段获取方法，一般不采用此方式，直接在前端字典缓存中获取即可 withEmptyFirst：如果需要在构造下集合数据源前增加一条空数据，需要设置此值为true，比如ListView的查询区域，数据字典瓦片样式查询前面增加查询“全部”项 isSpriteMethod：如果是调用自定义表单方法，设置此值为true method："post"或者"get"，远程方法调用的方式，默认为get params：远程方法调用参数，对象 source：远程方法调用url method_ids：同method，用在编辑视图中，根据存储的关联id还原远程调用关联的数据（数据库通常只存储Id，显示内容在打开编辑视图时，需要整体数据还原） params_ids: 同params source_ids: 同source sprite_text_eval：isSpriteMethod为true时生效，设置自定义表单方法调用后，下拉数据源显示值的动态配置 sprite_value_eval：isSpriteMethod为true时生效，设置自定义表单方法调用后，下拉数据源实际值的动态配置

1. 基础输入控件 控件名称：col-form-common，控件设置： `{'placeholder':'请输入string字段的数据','componentName':'a-input'}`
2. 日期控件 控件名称：col-form-common，控件设置：{'placeholder':'请输入datetime字段1的数据','componentName':'a-date-picker'}
3. 日期范围控件 控件名称：col-form-common，控件设置：{"componentName":"a-range-picker","valueFormat":"YYYY-MM-DD","style":"width: 200px","searchEventName":"change"}
4. 数据字典相关控件

- 控件名称：col-form-common，控件设置： `{"allowClear":true,"style":"width: 460px","componentName":"col-form-tile","sourceSettings":{"dict":"post","withEmptyFirst":true},"searchEventName":"change"}` 此例用在ListView普通查询区域，瓦片显示数据字典，在最开始增加“全部”字典项，控件选择改变时，触发列表查询
- 控件名称：SingleDropDown，控件设置： `{"placeholder":"请选择职务的数据","sourceSettings":{"dict":"post"}}` 此例子用在ListView高级查询区域，下拉单选数据字典
- 控件名称：SingleTile，控件设置： `{"placeholder":"请选择性别的数据","sourceSettings":{"params":{"dictCode":"sex"},"source":"/api/spriteform/common/GetDictItemRemoteInfo","method":"get"}}` 此例子用在ListView高级查询区域，下拉选择数据字典，数据字典的源为远程方法调用结果（数据字典一般不会这样用，这里只是举例远程调用构造下拉数据源）
- 控件名称：MultipleDropDown或者Checkbox，控件设置： `{"placeholder":"请选择性别的数据","sourceSettings":{"dict":"post"}}` 此例子用在ListView高级查询区域，下拉多选数据字典
- 控件名称：col-form-dict，控件设置： `{"placeholder":"请选择学历信息","dict":"edu","componentName":"a-select","mode":"multiple"}` 此例子用在ItemView的列控件设置处，下拉多选数据字典

5. 自定义表单方法调用 所有自定义表单方法调用都支持，比较通用的调用 控件名称：col-form-select，控件设置：

```
{"placeholder":"关联学生测试","allowClear":true,"sourceSettings":{"isSpriteMethod":true,"params":{"isTransaction":true,"methods":[{"methodId":'
```

此例子用在编辑视图的列控件，关联学生表，执行默认的PageList方法，模糊查询userName和name字段，返回的结果构造下拉选择数据源。 6. 系统用户选择控件 所有需要选择用户的地方都可以使用 用户选择控件特殊属性：

> isSelectByDept：是否支持弹出对话框选择用户 isForSearch：当用在高级查询时，设置为true withDeparture：设置是否查询离职用户

控件名称：col-form-user，控件设置：

```
{"placeholder":"请选择用户","allowClear":true,"style":"min-width: 170px","isSelectByDept":false,"searchEventName":"change"}
```

此例子用在ListView普通查询区域，选择用户之后立即执行查询事件 7. 下拉选择树控件 下拉选择树的应用场景，Object对象需要设置是否为树设置为true，系统自动添加并维护PId、TreeCode、Path、Icon、Title字段。 树控件控件名称设置为col-form-common，componentName设置为a-tree-select 调用后端TreeListWhere或者TreePageList方法，后端自动构造出树结构的对象，可以直接绑定到控件中。 设置默认展开第一级 treeDefaultExpandedFirst为true 例：

```
{"placeholder":"树查询","componentName":"a-tree-select","treeDefaultExpandedFirst":true,"style":"width: 170px","sourceSettings":{"isSpriteMeth
```

导入导出Excel模版

导入导出Excel组件设计文章，NOPI导入导出Excel组件封装 配置模型：

```
public class ExcelTemplate
   {
      /// <summary>
      /// 字段名称
      /// </summary>
      public string Field { get; set; }

      /// <summary>
      /// 列称
      /// </summary>
      public string Name { get; set; }

      /// <summary>
      /// 字段类型
      /// </summary>
      public EFieldType FieldType { get; set; }

      /// <summary>
      /// 列宽（显示多少个字符）
      /// </summary>
      public int CellLength { get; set; }

      /// <summary>
      /// 导出模版备注
      /// </summary>
      public string ExportComments { get; set; }

      /// <summary>
      /// 导入 是否必填
      /// </summary>
      public bool IsRequred { get; set; }

      /// <summary>
      /// 导入 验证类型
      /// </summary>
      public EValidateType ValidateType { get; set; }

      /// <summary>
      /// 导入 验证类型为String时，验证长度，为Regular，为正则表达式
      /// </summary>
      public string ValidateValue { get; set; }
```

```
    /// <summary>
        /// All = 1,OnlyForExport = 2(只在导入导出Excel使用),OnlyForImport = 3(只在导入导入Excel使用)
        /// </summary>
        public ETemplateForUse? TemplateForUse { get; set; }
    }
```

例：

```
"excelTemplate": [
    {
      "name": "字符",
      "field": "stringField",
      "fieldType": 5,
      "isRequred": true,
      "validateType": 0
    },
    {
      "name": "日期字段",
      "field": "dateTimeField",
      "fieldType": 7,
      "validateType": 10
    },
    {
      "name": "字典字段",
      "field": "dictField",
      "fieldType": 5,
      "validateType": 11
    },
    {
      "name": "数字",
      "field": "intField",
      "fieldType": 1,
      "validateType": 1
    },
    {
      "name": "长字符",
      "field": "textField",
      "fieldType": 5,
      "validateType": 0
    },
    {
      "name": "编号",
      "field": "autoNoField",
      "fieldType": 5,
      "templateForUse": 2,
      "validateType": 0
    },
    {
      "name": "浮点字段",
      "field": "decimalField",
      "fieldType": 2,
      "validateType": 3
    },
    {
      "name": "日期字段2",
      "field": "dateField",
      "fieldType": 6,
      "validateType": 9
    }
  ]
```

自定义查询条件

定义为一颗查询树，根据定义，动态拼接Sql查询语句，其中，value的值根据不同的查询条件动态变化的，相关的动态查询逻辑参考开源项目源码，具体请看如下示例： 例：

```
"sqlWheres": {
        "sqlExpressType": 1,
```

```
      "children": [
        {
          "sqlExpressType": 3,
          "field": "reason",
          "conditionType": 5,
          "value": "测试"
        },
        {
          "sqlExpressType": 3,
          "field": "flowStartTime",
          "conditionType": 3,
          "value": [
            "2022-04-20",
            "2022-04-21"
          ]
        }
      ]
    }
```

自定义查询数据模型

```
public class ExpressSqlModel
{
 public ESqlExpressType SqlExpressType { get; set; }
 public string Field { get; set; }
 public EConditionType ConditionType { get; set; }
 public object Value { get; set; }

 public List<ExpressSqlModel> Children { get; set; }
}

/// <summary>
/// Sql 表达式树
/// </summary>
public enum ESqlExpressType
{
 And = 1,
 Or = 2,
 Condition = 3
}

/// <summary>
/// Sql映射条件
/// </summary>
public enum EConditionType
{
 等于 = 1,
 不等于 = 2,
 Between = 3,
 In = 4,
 Like = 5,
 大于 = 6,
 大于等于 = 7,
 小于 = 8,
 小于等于 = 9,
 Null = 10,
 NotNull = 11,
 NotIn = 12
}
```

特殊时间查询

返回特定的开始时间和结束时间 特定枚举

```
public enum ETimeType
   {
      今天 = 1,
```

```
        本周 = 2,
        本月 = 3,
        本年 = 4,
        本周到今天 = 5,
        本月到今天 = 6,
        本年到今天 = 7,
        昨天 = 31,
        前天 = 32,
        前一周 = 33,
        上周 = 34,
        前一月 = 35,
        上月 = 36,
        前一年 = 37,
        去年 = 38,
        前年 = 39,
        前三天 = 40,
        明天 = 51,
        后天 = 52,
        下一周 = 53,
        下周 = 54,
        下一月 = 55,
        下月 = 56,
        下一年 = 57,
        明年 = 58,
        后年 = 59,
        后三天 = 60,
        所有 = 100,
        用户指定 = 101,
        更早 = 102,
        更晚 = 103,
    }
```

请求参数对象（在请求后端方法参数的datas字段中赋值）

```
internal class FormateTimeInfo
    {
        public ETimeType TimeType { get; set; }

        /// <summary>
        /// 计算的所在日期，为空，则计算当天
        /// </summary>
        public DateTime? CalculateDate { get; set; }

        /// <summary>
        /// 自定义时填写
        /// </summary>
        public DateTime? StartTime { get; set; }

        /// <summary>
        /// 自定义时填写
        /// </summary>
        public DateTime? EndTime { get; set; }
    }
```

配置帮助

1. 列表视图属性配置

列表视图属性设置

属性常规设置即为设置ant的table控件，另外附加扩展的一些自定义设置，ant的Table设置参考Table组件设置，其他一些扩展字段如下：

1. tableDiv 设置a-table外层的div样式，默认样式为：'min-height: 560px'
2. rowKey 前端表格的行主键，一般情况为"id"，可以不设置，当查询出来的列表数据主键不为id是，需要设置，如：id_a
3. tableType 自定义扩展字段，多数情况不需要设置，如果是树列表，需要设置为“tree”
4. columns 参见Table组件设置，定义表格列， 特殊处理（宽度width不设置，默认置为80；对齐align不设置，默认置为center，扩展linked属性，标识表格字段可以点击，点击事件可接入规则引擎；扩展option_is属性，标识字段列可自定义是否显示；扩展option_selected属性，配合option_is使用，标识字段列默认是否显示）
5. columnsExpanded columnsExpandedColSpan columnsExpanded表格展开更多内容字段列定义，columnsExpandedColSpan定义展开显示多少行，值为1-4，默认为4
6. colOperateWidth 定义列表操作列的宽度，当列表操作控件或者列表更多控件定义了，列表中的操作列才会显示。
7. excelTemplate 导入导出Excel配置模版信息，参见附录【导入导出Excel模版】，如：`[{"name":"电话","field":"phoneNumber","fieldType":5,"isRequred":true,"validateType":99,"validateValue":"[{'customerVal':'Phone','trigger':'blur'}]"},{"name":"学历","field":"education","fieldType":5,"validateType":11}]`
8. excelDicts 字段字典映射集合数据，表示Excel中用到的数据字典映射信息，定义哪个字段用到哪个数据字典，如：`[{"dict":"sex","field":"sex"},{"dict":"edu","field":"education"},{"dict":"title","field":"positionalTitle"}]`
9. excelName 导入导出Excel文件名称，如：人员管理测试Excel数据
10. uniqKey 导入时唯一字段检测（这里的字段为数据库字段，注意不是转换为驼峰命名之后的字段，多个字段组合用;号隔开），如：UserName。
11. eval_query 执行后端方法获取查询参数后执行的JS脚本，自定义扩展处理查询条件，本质上执行eval函数，特殊场景使用，比如执行查询之前，将查询条件做自定义特殊处理，如：界面查询条件只查询年月，到后端映射为时间段查询`sqlWhere.children.forEach(r=>{if(r.field==='checkTime'){r.value=[r.value.format('yyyy-MM')+'-01 00:00:00',r.value.add(1,'month').format('yyyy-MM')+'-01 00:00:00']}})`，参考附录：【自定义查询条件】
12. defaultSorting 默认后端查询方法排序，如：`checkTime desc,stockCheckType asc`等

高级功能说明

- 表头分页 参照antd设置即可，如果存在自定义列显示，且所有列都不显示时，分组字段也不显示

```
{
    "title": "日期字段",
    "children": [
      {
        "dataIndex": "dateTimeField",
        "title": "日期字段",
        "scopedSlots": {
          "customRender": "dateTimeField"
        },
        "width": 160
      },
      {
        "dataIndex": "dateField",
        "title": "日期字段2",
        "scopedSlots": {
          "customRender": "dateField"
        },
        "width": 160,
        "option_is": true,
        "option_selected": true
      }
    ]
  }
```

- 自定义列显示 在表格右上角，用户可点击下拉勾选自定义显示哪些字段列option_is属性控制是否允许用户选择，option_selected控制默认是否勾选

```
{
```

```
      "dataIndex": "dateField",
      "title": "日期字段2",
      "scopedSlots": {
        "customRender": "dateField"
      },
      "width": 160,
      "option_is": true,
      "option_selected": true
    }
```

- 是否换行显示 在表格右上角，checkbox选择，默认情况表格单元格内容超过展示宽度，用“...”代替，当勾选时，表格单元格内容换行显示所有内容。
- 超连接字段 扩展linked属性，点击单元格内容，可触发事件，接入规则引擎，触发的事件会将表格行作为事件参数传递到规则引擎，比如点击“合同编号”字段打开编辑合同对话框等（需要设置scopedSlots属性）。

```
{
    "align": "center",
    "dataIndex": "stringField",
    "title": "字符",
    "sorter": true,
    "linked": true,
    "scopedSlots": {
      "customRender": "stringField"
    }
  }
```

- 表格更多内容 columnsExpanded表格展开更多内容字段列定义，columnsExpandedColSpan定义展开显示多少行，值为1-4，默认为4，表格更多列不支持option_is和option_selected属性，支持linked属性设置，“是否换行显示”功能也支持
- 超出内容自动横向滚动 表格控件设置了超出表格所有列长度时，自动出现横向滚动条
- 其他高级功能 更多高级功能，可增加列表自定义控件，格式化显示表格内容