

# TaskBuilder

## 低代码开发入门

北京任讯信息技术有限公司

[www.taskmsg.com](http://www.taskmsg.com)

## 目 录

|                                    |    |
|------------------------------------|----|
| 一、 TaskBuilder 简介 .....            | 1  |
| 1、TaskBuilder 的定位 .....            | 1  |
| 2、TaskBuilder 目标用户 .....           | 2  |
| 3、TaskBuilder 适用场景 .....           | 3  |
| 4、TaskBuilder 技术架构 .....           | 3  |
| 5、TaskBuilder 功能特性 .....           | 4  |
| 5.1 基础功能组件化 .....                  | 4  |
| 5.2 功能设计可视化 .....                  | 5  |
| 5.3 应用创建向导化 .....                  | 7  |
| 5.4 前后端分离松耦合 .....                 | 9  |
| 5.5 多端适配一体化 .....                  | 9  |
| 5.6 一种语言全栈开发 .....                 | 10 |
| 5.7 应用代码中性化 .....                  | 10 |
| 5.8 功能扩展开放化 .....                  | 11 |
| 6、TaskBuilder 支持的三种开发模式 .....      | 11 |
| 6.1 多层分离的低代码开发模式 .....             | 11 |
| 6.2 基于数据表单的零代码开发模式 .....           | 12 |
| 6.3 基于配置信息的一键生成式开发模式 .....         | 14 |
| 7、任讯应用开发平台的构成 .....                | 14 |
| 7.1 任擎 Tasgine 应用服务引擎 .....        | 15 |
| 7.2 任流 TaskFlow  workflow 引擎 ..... | 20 |

|                               |    |
|-------------------------------|----|
| 7.3 任讯 TaskMsg 即时通讯工具 .....   | 25 |
| 7.4 任集 TaskStore 应用市场 .....   | 30 |
| 二、TaskBuilder 开发平台安装配置 .....  | 31 |
| 1、任擎服务器 Windows 版安装配置 .....   | 31 |
| 第一步：下载任擎服务器 .....             | 31 |
| 第二步：使用安装向导配置任擎服务器 .....       | 31 |
| 2、任擎服务器 CentOS 版安装配置指南 .....  | 36 |
| 第一步：下载并初始化任擎服务器 .....         | 36 |
| 第二步：配置任擎服务器 .....             | 36 |
| 常用命令 .....                    | 37 |
| 3、任擎服务器 macOS 版安装配置指南 .....   | 38 |
| 第一步：下载并初始化任擎服务器 .....         | 38 |
| 第二步：配置任擎服务器 .....             | 38 |
| 第三步：启动任擎服务器 .....             | 39 |
| 停止任擎 .....                    | 39 |
| 4、任擎服务器 Docker 版安装配置指南 .....  | 40 |
| 第一步：准备宿主机运行环境 .....           | 40 |
| 第二步：安装配置任擎版 MySQL 数据库容器 ..... | 40 |
| 第三步：设置任擎服务器配置文件 .....         | 41 |
| 第四步：安装配置任擎服务器容器 .....         | 41 |
| 三、TaskBuilder 开发环境简介 .....    | 42 |
| 1、TaskBuilder 主界面 .....       | 42 |

|                                 |    |
|---------------------------------|----|
| 2、系统侧边栏 .....                   | 43 |
| 2.1 数据源管理 .....                 | 43 |
| 2.2 项目资源管理 .....                | 45 |
| 2.3 文件同步工具 .....                | 55 |
| 2.4 SQL 执行工具 .....              | 56 |
| 2.5 后台服务测试工具 .....              | 57 |
| 2.6 登录后台管理系统 .....              | 58 |
| 2.7 进入开发者社区 .....               | 58 |
| 3、服务器设置 .....                   | 59 |
| 4、系统辅助工具 .....                  | 61 |
| 5、系统标签栏 .....                   | 62 |
| 6、系统状态栏 .....                   | 62 |
| 四、TaskBuilder 低代码开发基础知识介绍 ..... | 63 |
| 1、数据库相关知识 .....                 | 63 |
| 1.1 关系型数据库简介 .....              | 63 |
| 1.2 为什么要用数据库? .....             | 64 |
| 1.3 数据库中的表和视图 .....             | 65 |
| 1.4 SQL 语言 .....                | 67 |
| 2、JavaScript 语言 .....           | 68 |
| 2.1 JavaScript 语言简介 .....       | 68 |
| 2.2 JavaScript 语言的特点 .....      | 69 |
| 2.3 JavaScript 语言的标准 .....      | 69 |



|                               |    |
|-------------------------------|----|
| 2.4 JavaScript 语言的模块化规范 ..... | 70 |
| 2.5 JavaScript 语言的运行环境 .....  | 70 |
| 2.6 JSON 数据格式 .....           | 71 |
| 3、HTML 语言 .....               | 73 |
| 3.1 HTML 简介 .....             | 73 |
| 3.2 HTML 的版本 .....            | 74 |
| 3.3 HTML 文档的结构 .....          | 75 |
| 4、CSS 样式 .....                | 76 |
| 4.1 CSS 简介 .....              | 76 |
| 4.2 CSS 的特点 .....             | 77 |
| 4.3 CSS 语法 .....              | 78 |
| 4.4 CSS 选择器 .....             | 79 |
| 4.5 CSS 的使用 .....             | 81 |
| 5、Node.js .....               | 83 |
| 5.1 Node.js 简介 .....          | 83 |
| 5.2 Node.js 的模块化 .....        | 84 |
| 5.3 npm 包管理工具 .....           | 87 |
| 6、HTTP 协议 .....               | 88 |
| 6.1 HTTP 简介 .....             | 88 |
| 6.2 HTTP 工作原理 .....           | 89 |
| 6.3 HTTP 的特点 .....            | 90 |
| 6.4 常用 HTTP 方法 .....          | 91 |

|                                 |     |
|---------------------------------|-----|
| 6.5 HTTP 响应状态码 .....            | 93  |
| 6.6 HTTPS 协议 .....              | 93  |
| 7、URL 格式 .....                  | 95  |
| 7.1 URL 简介 .....                | 95  |
| 7.2 URL 语法 .....                | 95  |
| 7.3 URL 格式说明 .....              | 95  |
| 五、TaskBuilder 低代码开发初体验 .....    | 97  |
| 1、示例应用简介 .....                  | 97  |
| 2、创建学员管理项目 .....                | 98  |
| 3、创建学员信息数据模型 .....              | 99  |
| 4、创建学员信息列表页面 .....              | 102 |
| 5、创建学员信息添加页面 .....              | 107 |
| 6、创建学员信息修改页面 .....              | 110 |
| 7、预览和测试 .....                   | 113 |
| 8、发布应用 .....                    | 116 |
| 六、TaskBuilder 低代码开发基本原理解析 ..... | 121 |
| 1、信息系统常见的系统架构 .....             | 121 |
| 1.1 单文件架构 .....                 | 121 |
| 1.2 两层架构 .....                  | 122 |
| 1.3 多层架构 .....                  | 122 |
| 1.4 TaskBuilder 支持的系统架构 .....   | 124 |
| 2、前端文件和后台服务的概念 .....            | 125 |

|                                  |     |
|----------------------------------|-----|
| 2.1 前端文件 .....                   | 125 |
| 2.2 后台服务 .....                   | 125 |
| 3、TaskBuilder 低代码开发必须的三种文件 ..... | 126 |
| 3.1 数据模型文件 .....                 | 126 |
| 3.2 后台服务文件 .....                 | 130 |
| 3.3 前端页面文件 .....                 | 140 |
| 4、前后端数据交互流程解析 .....              | 156 |
| 4.1 客户端请求前端静态文件的流程 .....         | 156 |
| 4.2 前端请求后台服务的方式 .....            | 158 |
| 4.3 触发前端组件请求后台服务的常见事件 .....      | 166 |
| 4.4 客户端请求后台服务的流程 .....           | 168 |
| 4.5 前后端通讯的数据格式 .....             | 169 |
| 4.6 常见页面前后端数据交互原理解析 .....        | 171 |
| 七、TaskBuilder 低代码开发项目实战 .....    | 196 |
| 1、实战项目简介 .....                   | 196 |
| 2、创建项目 .....                     | 196 |
| 3、客户管理模块开发 .....                 | 198 |
| 3.1 本节知识点 .....                  | 198 |
| 3.2 创建客户信息数据模型 .....             | 199 |
| 3.3 创建客户信息文件夹 .....              | 210 |
| 3.4 创建客户信息列表页面 .....             | 212 |
| 3.5 如何查看前后端错误和调试信息 .....         | 231 |

|                          |     |
|--------------------------|-----|
| 3.6 创建客户信息添加页面 .....     | 242 |
| 3.7 抽屉页使用简介 .....        | 251 |
| 3.8 完善客户信息列表页面 .....     | 257 |
| 3.9 创建客户信息修改页面 .....     | 268 |
| 3.10 客户信息修改页实现原理解析 ..... | 274 |
| 3.11 发布客户管理应用 .....      | 279 |
| 4、产品管理模块开发 .....         | 292 |
| 4.1 本节知识点 .....          | 292 |
| 4.2 创建数据模型 .....         | 292 |
| 4.3 创建产品信息文件夹 .....      | 294 |
| 4.4 创建产品信息列表页面 .....     | 296 |
| 4.5 实现产品分类树形展示 .....     | 302 |
| 4.6 实现添加产品分类的功能 .....    | 309 |
| 4.7 实现修改产品分类的功能 .....    | 328 |
| 4.8 实现删除产品分类的功能 .....    | 341 |
| 4.9 实现添加产品信息的功能 .....    | 343 |
| 4.10 实现修改产品信息的功能 .....   | 357 |
| 4.11 完善产品信息的删除功能 .....   | 367 |
| 4.12 实现产品信息禁用和启用 .....   | 369 |
| 4.13 实现产品信息导入和导出 .....   | 376 |
| 4.14 发布产品管理应用 .....      | 390 |
| 5、销售订单模块开发 .....         | 390 |

|                           |     |
|---------------------------|-----|
| 5.1 本节知识点 .....           | 390 |
| 5.2 创建数据模型 .....          | 391 |
| 5.3 创建销售管理文件夹 .....       | 393 |
| 5.4 创建销售订单列表页面 .....      | 394 |
| 5.5 设置销售订单编码规则 .....      | 399 |
| 5.6 创建添加销售订单页面 .....      | 403 |
| 5.7 完善添加销售订单页面 .....      | 405 |
| 5.8 预览测试添加销售订单页面 .....    | 429 |
| 5.9 创建修改销售订单页面 .....      | 430 |
| 5.10 实现查看销售订单详情功能 .....   | 440 |
| 5.11 实现销售订单审核功能 .....     | 444 |
| 5.12 实现订单列表按状态显示 .....    | 450 |
| 5.13 设置列表数据懒加载 .....      | 459 |
| 5.14 发布销售订单管理应用 .....     | 461 |
| 6、销售统计模块开发 .....          | 461 |
| 6.1 本节知识点 .....           | 461 |
| 6.2 创建数据统计所需的视图 .....     | 462 |
| 6.3 实现按客户统计销售情况 .....     | 467 |
| 6.4 实现按产品统计销售情况 .....     | 474 |
| 6.5 实现按日期统计产品分类销售情况 ..... | 482 |
| 6.6 用统计图显示产品销售情况 .....    | 491 |
| 6.7 实现销售情况按月统计 .....      | 495 |

# 一、TaskBuilder 简介

## 1、TaskBuilder 的定位

随着这两年“低代码”概念的火爆，现在市面上出现了很多所谓的零代码和低代码开发平台，其中有大厂商建设的云平台级低代码，涵盖的范围很广，也有专注某些领域的低代码，例如建站、小程序开发、APP 开发、可视化大屏、数字孪生、物联网组态等等，在开发管理信息系统这个方向，又分为简易表单流程驱动的零代码、网络表格类低代码、OA 或 BPM 厂商的低代码、ERP 厂商的低代码等多个流派，这些低代码都有各自的优缺点，也有各自的适用场景和目标用户，具体对比如下表所示：

| 类型              | 优点                                          | 缺点                           | 适用场景           | 适用用户         |
|-----------------|---------------------------------------------|------------------------------|----------------|--------------|
| 简易表单流程驱动的零代码    | 简单易用                                        | 不能开发复杂业务系统<br>云端部署,不支持私有化    | 简单的工作<br>申请和审批 | 办公人员         |
| 网络表格类低代码        | 比多层分离的低代码简单, 功能比零代码强大                       | 仅能开发中等复杂度的系统, 流程功能弱          | 简单的数据<br>填报和查询 | IT 人员        |
| OA 或 BPM 厂商的低代码 | 功能比零代码强大, 能实现复杂的业务流程处理                      | 功能有局限, 不适合开发非流程类业务系统         | 复杂的工作<br>申请和审批 | IT 人员        |
| ERP 厂商的低代码      | 功能强大, 尤其是开发 ERP 类业务                         | 操作复杂, 主要跟自家 ERP 适配, 不够开放, 贵! | ERP 系统<br>功能扩展 | IT 人员        |
| 多层分离的低代码        | 数据模型与业务分离, 前端界面和后台服务分离, 界面与逻辑分离, 功能强大、灵活可扩展 | 操作相对较复杂<br>学习成本高             | 专业管理信<br>息系统   | IT 人员<br>程序员 |

从广义上讲，上面的各种应用场景，使用整个任讯应用开发平台提供的丰富功能都可以解决，但单从 TaskBuilder 来说，它最有特色的还是“多层分离”的低代码开发模式，虽然 TaskBuilder 也有支持零代码开发的“数据表单”功能，但这不是重点，也不是特点。我们研发 TaskBuilder 的初衷，不只是想让它仅能开发一些简单的应用，而是能用它快速开发专业的、复杂的信息系统，所以 TaskBuilder 的目标是既要尽可能地简单易用，又要功能强大、灵活可扩展。为此，TaskBuilder 在底层架构层面，就进行了精心的设计，既可以通过组件化、向导化、可视化、模版化等多种手段，让软件开发变得更加简单，又通过采取多层分离的架构，将数据模型与业务分离、前端界面和后台服务分离、界面与逻辑分离，每一层都提供了功能强大的开发工具，可以进行细致精确的设计，而不是混杂在一起，遇到复杂的业务就无法灵活处理。而且，在此基础上，为了最大程度地简化开发，又提供了基于“数据表单”和“快应用”这两种形式的零代码开发模式，可以满足不同技术基础的用户，大家可以根据自身需求进行选择。

另外，TaskBuilder 提供免费共享的版本，没有任何限制，支持私有化部署，也是很多低代码开发者选择 TaskBuilder 的重要原因。并且我们投入大量精力建设和维护开发者社区和技术支持群，为广大开发者提供及时专业的业务咨询、问题处理、学习文档和视频等服务，积极倾听每位开发者的意见和建议，坚持每月发布一版，持续优化和完善，目前核心功能已经非常稳定，从 2022 年 2 月正式发布以来，累计下载量已突破 30 万，成为国内低代码领域有一定影响力的品牌，并受邀参与低代码开发平台国家标准的编制。

## 2、TaskBuilder 目标用户

- **零编程基础人员：**如果只是开发简单的增删改查应用，不需要了解任何编程知识，只要会使用 office 软件，经过简单的培训，就可以通过向导一步一步操作开发出具体的业务功能。
- **IT 技术支持人员：**如果是有一定技术基础的 IT 技术支持人员，例如 ERP、OA、CRM、HR 等软件的实施人员，企业内部的信息系统维护人员等，会简单的 js 脚本和 sql 语言，则可以开发更加复杂的应用。
- **程序员：**如果是专业的软件开发人员，则只需要掌握 html+js+css 和 sql 这几种技术即可开发大型的、全功能的复杂业务系统。

### 3、TaskBuilder 适用场景

- **开发增删改查 (CRUD) 应用：**TaskBuilder 主要用来快速开发数据库增删改查 (CRUD) 类应用，包括政府和企事业单位内部的各种行政和业务管理系统等，支持电脑端和手机端 (H5)，也支持微信小程序。
- **开发工作流程类应用：**与任讯第四代工作流引擎 TaskFlow (任流) 配合，可以实现各种复杂业务流程的发起、流转、监控和查询统计。
- **设计项目静态原型：**也可以用 TaskBuilder 设计系统界面原型，以便跟客户确认需求或内部交流讨论。
- **集成各类现有系统：**TaskBuilder 可以通过应用集成、API 集成和数据库集成等多种方式，实现内部各个业务系统之间的整合集成，建立统一的信息门户。

### 4、TaskBuilder 技术架构

TaskBuilder 主要用来开发电脑端 Web 应用和手机端 H5 应用，所以前端采用的技术就是 HTML+JavaScript+CSS，前端 UI 框架采用的我们自主研发的开源 UI 框架 tfp (TaskMsg Front Page)。

后端默认采用我们基于 Node.js 自主研发的应用服务引擎 Tasgine (任擎)，后台服务的默认开发语言为 JavaScript，可以做到掌握一种编程语言 (JavaScript) 即可实现前后端全栈开发。

后端也可以用 Java、C#、PHP 等其他开发语言，只要符合 tfp 组件的前后端数据通讯格式即可，但也需要使用 Tasgine (任擎) 作为 tfp 页面的渲染服务器和后台服务的代理服务器，或者采用支持微服务架构的集群版任擎，具体实现方法见 Tasgine (任擎) 服务器的介绍文档。

如果后台使用其他开发语言，则只能用 TaskBuilder 进行前端界面设计和脚本编辑，暂时无法用 TaskBuilder 开发其他语言的后台服务，需要使用其他开发工具开发。



## 5、TaskBuilder 功能特性

### 5.1 基础功能组件化

TaskBuilder 将常用的功能封装成了组件，包括前端 UI 组件、后台业务操作等，开发业务功能时，可以像搭积木一样，通过鼠标拖拽就能快速实现前端界面设计和后台功能开发。



tfp 前端 UI 组件



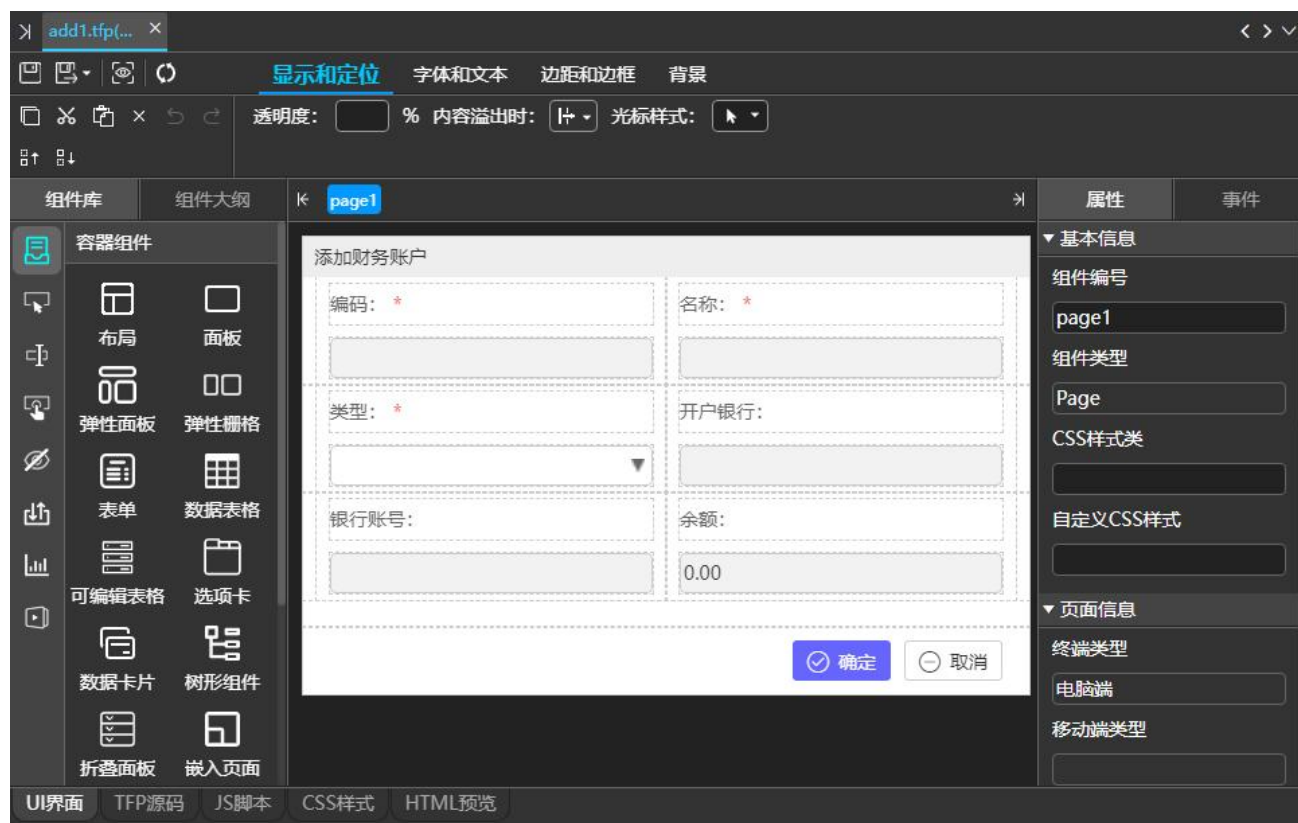
组件整体结构

5.2 功能设计可视化

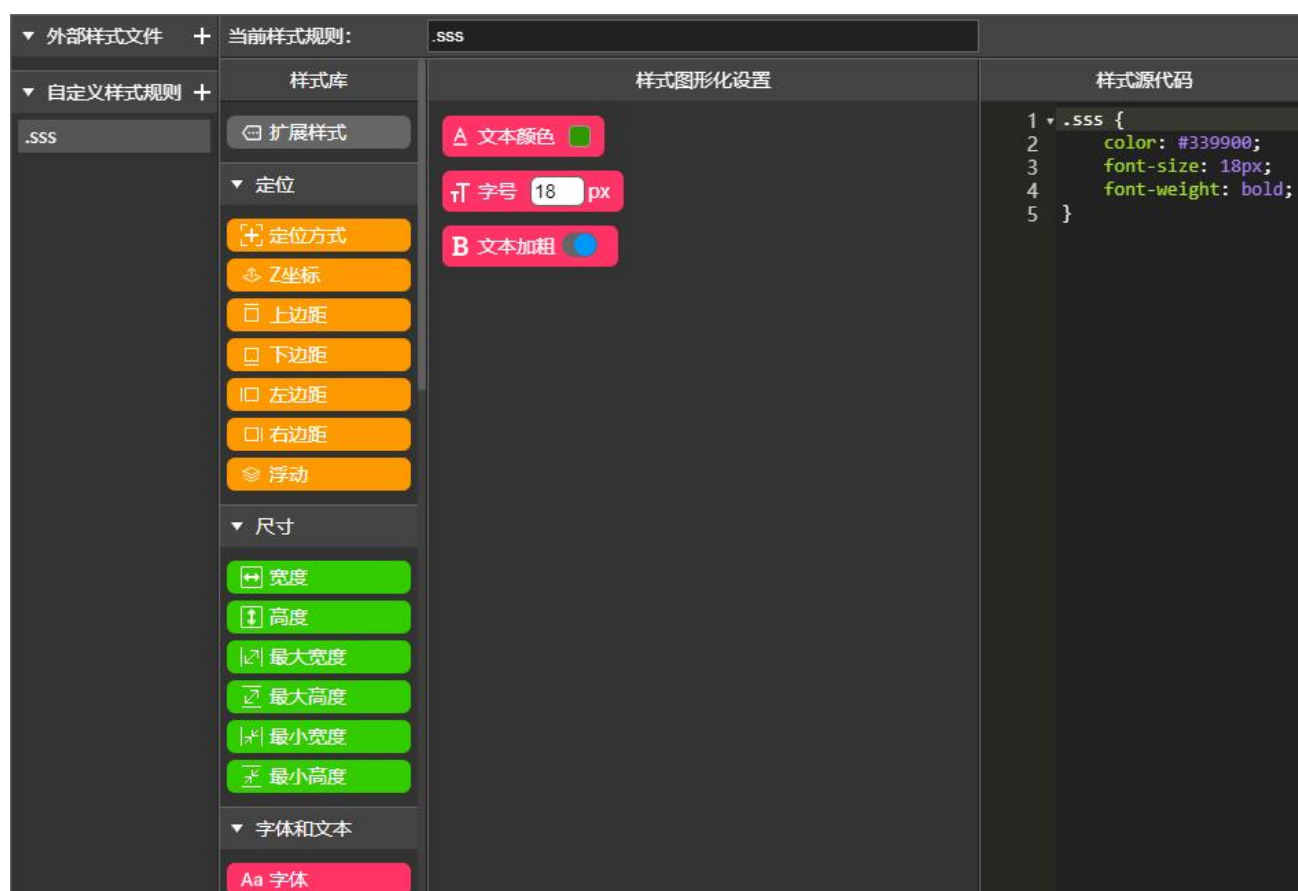
使用 TaskBuilder 开发应用时，基本上大多数操作都可以使用图形化的工具实现，包括数据结构设计、界面设计、样式设置、业务逻辑设置等，可以尽量减少代码的编写。



数据模型设计器



UI 界面设计器



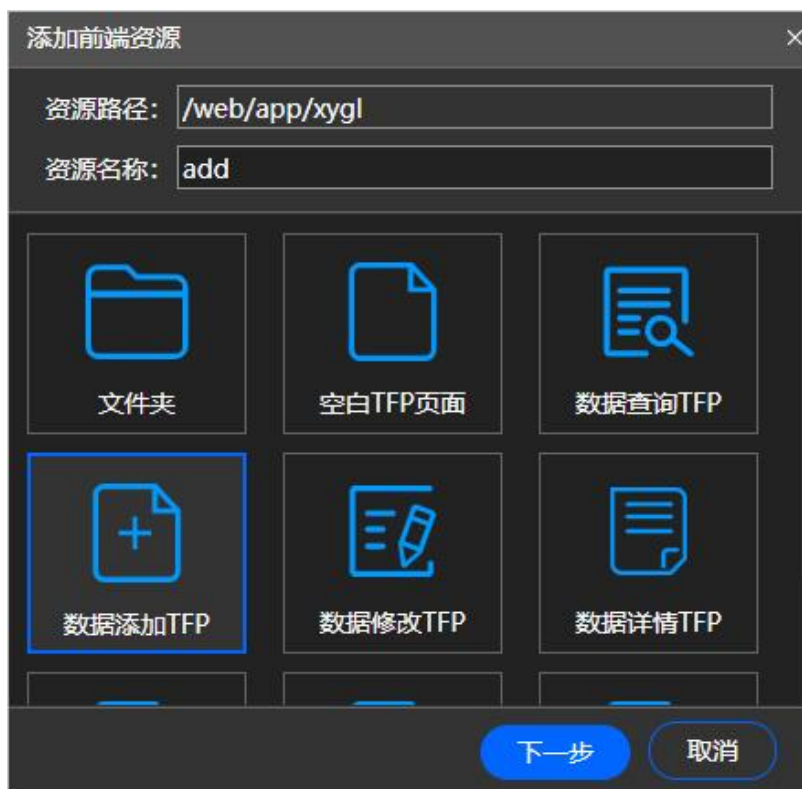
CSS 样式设计器



业务逻辑设计器

## 5.3 应用创建向导化

TaskBuilder 提供了丰富的开发向导，用户按照向导一步一步操作，就可以快速创建增删改查应用（CRUD）。



选择页面类型



创建数据添加TFP

保存路径: /web/app/xygl/add.tfp

数据模型: xygl/model/xyxx.tdm

选择...

页面标题: 添加学员信息

页面类型: 抽屉页

窗口大小: 宽 高

终端类型: 电脑端

背景颜色: 浅色

定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧

每行显示输入项数量: 3 仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务

选择...

| <input checked="" type="checkbox"/> | 字段    | 输入项名称 | 输入项类型 | 宽度 | 占用列数 | 必填                                  | 可选值        | 数据字典(双击选择) | 默认值 |
|-------------------------------------|-------|-------|-------|----|------|-------------------------------------|------------|------------|-----|
| <input checked="" type="checkbox"/> | xb    | 性别    | 单选框   |    |      | <input checked="" type="checkbox"/> | 男,女        |            | 男   |
| <input checked="" type="checkbox"/> | sfzh  | 身份证号  | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | whcd  | 文化程度  | 下拉列表  |    |      | <input checked="" type="checkbox"/> | 小学,初中,高中,大 |            |     |
| <input checked="" type="checkbox"/> | xz    | 乡镇    | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | xzc   | 行政村   | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | sfpkh | 是否贫困户 | 开关    |    |      | <input checked="" type="checkbox"/> | 是否         |            | 否   |
| <input checked="" type="checkbox"/> | dhhm  | 电话号码  | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | bz    | 备注    | 多行输入框 |    | 3    | <input type="checkbox"/>            |            |            |     |

确定

取消

设置相关参数

显示和定位 字体和文本 边距和边框 背景

透明度: % 内容溢出时: 光标样式:

组件库 组件大纲 page1

容器组件

布局 面板 弹性面板 弹性栅格 表单 数据表格 可编辑表格 选项卡 数据卡片 树形组件 折叠面板 嵌入页面 宫格 宫格行 宫格列 单元

培训年份: \* 期数: \* 班次: \*

姓名: \* 性别: \* 身份证号: \*

文化程度: \* 乡镇: \* 行政村: \*

是否贫困户: \* 电话号码: \*

备注:

属性 事件

基本信息

组件编号 page1 组件类型 Page CSS样式类 自定义CSS样式

页面信息

终端类型 电脑端 移动端类型 组件默认定位方式 浮动定位 前端UI框架 tfp 页面类型

UI界面 TFP源码 JS脚本 CSS样式 HTML预览

完成页面创建

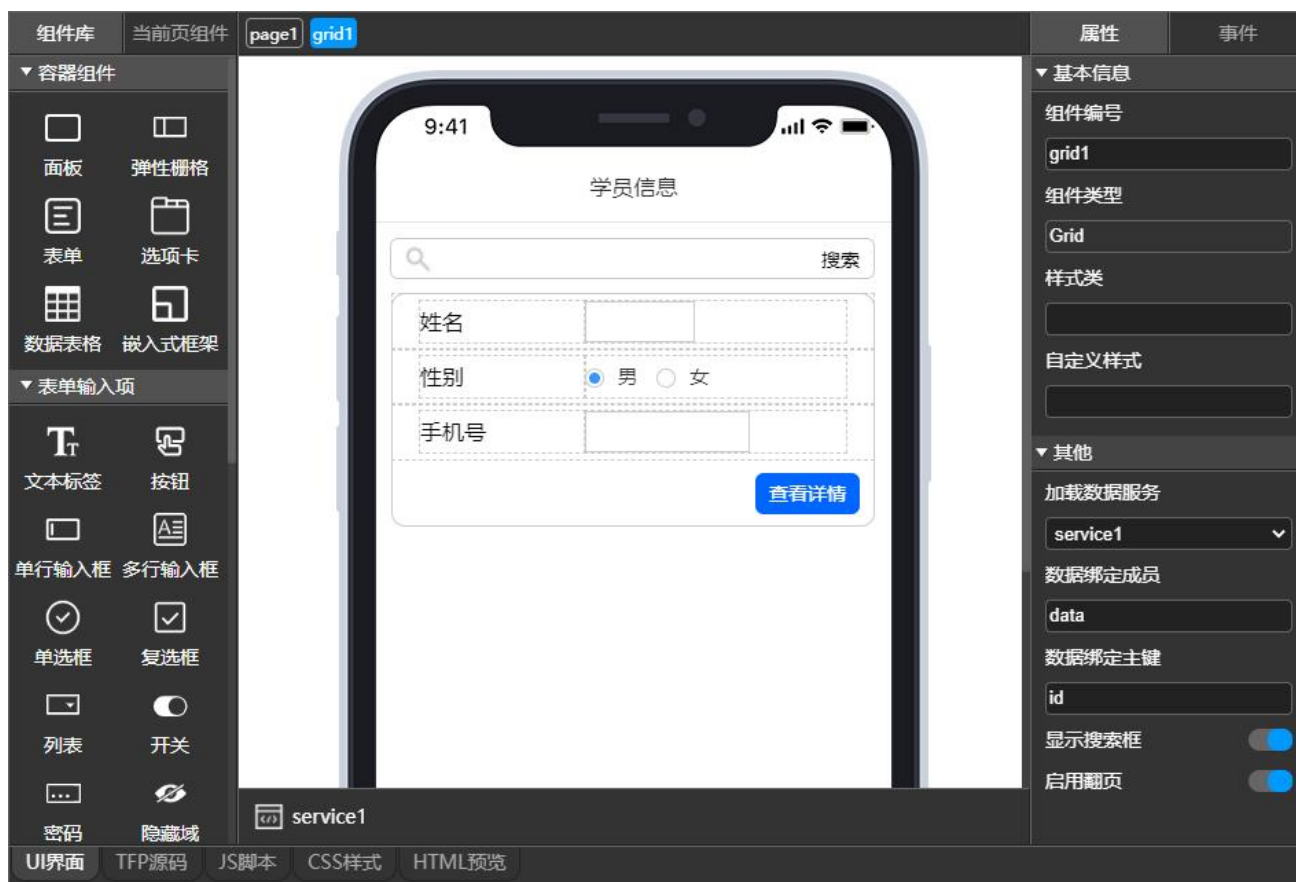
## 5.4 前后端分离松耦合

使用 TaskBuilder 进行低代码开发时，前端界面和后台服务代码不是混杂在一起、强依赖的，是彻底分开的，采用 JSON 格式进行数据传输，一套前端界面可以支持多种后端语言，一套服务也可以给多套界面使用，只要传输的数据格式满足要求即可。

而且，前端界面设计和前端业务逻辑代码也是分离的，易于设计和维护。简单的业务功能，通过可视化拖拽设计即可完成，如果前端有复杂的业务逻辑，可以在独立的代码编辑器内编写前端脚本，所有前端组件都封装成了 JavaScript 对象，在前端脚本内，可以用面向对象的形式获取或设置组件的属性，调用组件方法，处理组件事件。

## 5.5 多端适配一体化

TaskBuilder 目前已支持开发电脑端 Web 应用、手机端 H5 应用和微信小程序，手机端 H5 应用还支持嵌入到任讯、阿里钉钉和企业微信等多种 APP 中。



H5 页面设计

## 5.6 一种语言全栈开发

使用 TaskBuilder 开发应用时，前后端都可以采用 JavaScript 这一种编程语言，不需要学习多种编程语言。

## 5.7 应用代码中性化

用 TaskBuilder 开发的前端页面，保存后的内容格式为 JSON 格式，扩展名为.tfp，当有用户访问该页面时，引擎会自动将其编译为 HTML 页面，以便浏览器解析执行，包括页面里的所有 tfp 组件所需的 js 代码和 css 代码，都会统一打包编译到这个 HTML 页面内。目前，tfp 组件采用的前端框架为 jQuery，将来可能会支持 VUE、React 等其他框架，到时，如果要更换前端框架，整个页面不需要重新开发，只需要修改一下 tfp 页面源码中的框架类型即可，到时，引擎会自动编译成采用新框架的页面代码。

```
1  {
2      "client": "pc",
3      "framework": "tfp",
4      "positionType": "float",
5      "bgColorMode": "light",
6      "pageType": "drawer",
7      "title": "添加客户信息",
8      "type": "Page",
9      "docType": "HTML5",
10     "components": [
11         {
12             "id": "panel_page",
13             "type": "Panel",
14             "components": [
15                 {
16                     "id": "form1",
17                     "type": "Form",
18                     "styles": {
19                         "left": "0",
20                         "top": "10px",
```

tfp 前端页面源码

使用 TaskBuilder 开发的后台服务，默认保存的格式也是 JSON 格式，扩展名为.tbs，引擎会自动编译为 JavaScript 代码。这种中性的 JSON 格式，理论上可以编译为任何其他编程语言，例如 Java、C#和 PHP 等，只是暂时还不支持，将来会陆续支持。

```
1  {
2    "name": "Add",
3    "comment": "添加ledger_list",
4  > "reqArgs": [ ...
40  ],
41  > "retArgs": [ ...
46  ],
47  "methods": [
48    {
49      "name": "process",
50      "comment": "处理服务请求",
51      "args": [
52        {
53          "name": "req",
54          "type": "object",
55          "comment": "服务请求对象"
56        },
57        {
58          "name": "res",
59          "type": "object",
60          "comment": "服务响应对象"
61        }
57      ]
58    }
59  ]
60 }
61 }
```

tbs 后台服务源码

## 5.8 功能扩展开放化

TaskBuilder 的代码是开源的，开发人员可以根据自身需求按需扩展，包括功能、组件、模板、向导等，都可以随时扩展。

# 6、TaskBuilder 支持的三种开发模式

## 6.1 多层分离的低代码开发模式

TaskBuilder 最有特色的开发模式就是多层分离的低代码开发模式，该模式既可以通过组件化、图形化、向导化等多种手段大幅降低应用开发的难度，相对纯代码开发来说，要容易很多，但由于采取了多层分离的模式，每一层都可以单独进行精细化设置，而且前后端逻辑既可以图形化编排，也可以直接写码，或引入第三方代码，可以随时扩展系统功能，所以，这种模式功能是最强大的，纯代码能做到的，TaskBuilder 同样也能做到，可以说 TaskBuilder 将简



单易用和强大灵活这两个本应相互对立的两个特性巧妙地融合到了一起, 相对其他零代码或低代码开发工具来说, 能适用于更多的应用场景, 尤其是开发一些复杂的、专业的信息化系统, 会更有优势。

TaskBuilder 的多层分离包括以下几个方面:

- **数据模型与前后端分离:** 可以单独定义与数据库中的表或视图对应的数据模型文件, 一个数据模型文件可以供多个前后端文件使用, 而不是像基于表单的零代码开发模式那样, 把界面和数据结构混杂在一起, 不能复用。另外, 这种模式还可以继续利用企业现有各个应用系统数据库中的数据, 不需要再设计新的表单, 只需要导出数据结构即可。
- **前端页面与后台服务分离:** 前端页面和后台服务都保存在独立的文件中, 可以分开设置, 前端页面负责填报和展示数据, 和用户交互, 后台服务负责对数据库进行增删改查操作。一个前端页面可以调用多个后台服务, 一个后台服务可以供多个前端页面使用。
- **前端界面和前端逻辑分离:** 界面 UI 设计和界面业务逻辑编排分离, 各有各的编辑区域, 界面 UI 完全可视化设计, 可以精确设置每个组件的属性和样式, 界面业务逻辑可以图形化编排, 也可以写码, 再复杂的业务逻辑都可以实现。

采取这种多层分离的模式, 有以下好处:

- 各个功能分工更加明确清晰, 而不是混杂在一起, 各个部分可以独立查看和维护;
- 资源可以复用, 一个数据模型可以给多个应用使用, 一个后台服务可以给多个前端页面使用, 不需要重复开发配置;
- 各个部分松耦合, 如果某一层要更换技术栈, 其他部分不需要动, 例如后台服务的语言改成 Java 或 C#, 前端页面和数据模型都不用动, 或者数据模型对应的数据库由 MySQL 改查 SQLServer, 也只需要对数据模型做一些适当的修改, 不需要整个应用都改一遍。

多层分离的低代码开发模式是 TaskBuilder 其他开发模式的基础, 其他模式不管操作界面和方式如何, 最后实际上都会生成“数据模型+后台服务+前端页面”这三种核心元素, 如果需要, 可以采取低代码模式继续编辑完善。

本文档仅介绍这种多层分离的低代码开发模型, 其他开发模式会有单独的文档进行介绍。

## 6.2 基于数据表单的零代码开发模式

在 TaskBuilder 内, 最简单的开发模式就是基于数据表单的零代码开发模式, 这种模式将

数据模型、后台服务和前端页面这三种元素统一合并到一个叫“数据表单”的文件里，只需要在这一个文件中配置填报和查询界面，就会自动生成对应的数据模型、后台服务和前端页面，而且界面设置非常简单易用，不需要任何编程基础，直接就可以上手操作。但相应地，该模式为了尽量简化操作，不支持配置复杂的前后端业务逻辑，也不支持自定义组件样式，所以，这种开发模式只适合开发一些简单的应用，不支持开发特别复杂的业务系统。



The image shows the 'Form Design' (填报设计) interface. It features a top navigation bar with tabs for '填报设计', '查询设计', '手机填报设计', and '手机查询设计'. On the right, there are buttons for '保存' (Save), '预览' (Preview), '刷新' (Refresh), and '更多' (More). The main area is divided into three panels: '基本' (Basic) on the left with various input controls like '单行输入', '下拉框', '单选框', etc.; '表单设计' (Form Design) in the center for configuring fields like '名称' (Name), '代码' (Code), '开关' (Switch), '多行输入' (Multi-line input), '下拉框' (Dropdown), and '开关'; and '控件编码' (Control Code) on the right for setting properties like '控件名称' (Control Name), '宽度' (Width), '方向' (Direction), '必填' (Required), and '只读' (Read-only).

填报界面设计



The image shows the 'List Design' (列表设计) interface. It has a top navigation bar with tabs for '填报设计', '查询设计', '手机填报设计', and '手机查询设计'. On the right, there are buttons for '保存' (Save), '预览' (Preview), '刷新' (Refresh), and '更多' (More). The main area is divided into three panels: '全部' (All) on the left with a '添加' (Add) button and a table with columns '名称' (Name) and '代码' (Code); '列表设置' (List Settings) in the center with options for '表格设置' (Table Settings), '显示列' (Display Columns), '数据过滤' (Data Filter), '排序' (Sort), and '搜索' (Search); and '显示列' (Display Columns) on the right for configuring the list columns. Below the '显示列' panel is a table for '表单字段' (Form Fields).

|                                     | 字段   | 对齐 | 宽度 | 合计                       |     |
|-------------------------------------|------|----|----|--------------------------|-----|
| <input checked="" type="checkbox"/> | 名称   | 居中 | 默认 | <input type="checkbox"/> | ^ v |
| <input checked="" type="checkbox"/> | 代码   | 居中 | 默认 | <input type="checkbox"/> | ^ v |
| <input checked="" type="checkbox"/> | 开关   | 居中 | 默认 | <input type="checkbox"/> | ^ v |
| <input checked="" type="checkbox"/> | 多行输入 | 居中 | 默认 | <input type="checkbox"/> | ^ v |
| <input checked="" type="checkbox"/> | 下拉框  | 居中 | 默认 | <input type="checkbox"/> | ^ v |
| <input checked="" type="checkbox"/> | 开关   | 居中 | 默认 | <input type="checkbox"/> | ^ v |

列表界面设计

## 6.3 基于配置信息的一键生成式开发模式

这种开发模式称为“快应用”，它的操作难度介于上面的两种模式之间，比低代码简单，比零代码复杂；功能比零代码强大，比低代码弱一些。适合用零代码实现不了、但用低代码又觉得复杂的场景。这种模式以应用配置信息为核心，可以给一个应用配置多种类型的页面，包括列表页面、填报和查看页面等，每种类型的页面都可以配置多个。列表页里可以配置查询条件、要查询的字段、列表里显示的列及详细参数、排序条件、分组条件、可以进行哪些操作等，填报页面可以配置要显示的字段及其输入项。对这些页面的配置类似使用前端页面创建向导创建页面时所做的设置一样，区别是向导只能一次性使用，创建完页面后就没法再使用向导里的那些功能来配置页面，而基于配置信息的开发模块可以随时修改配置，改完点击生成就能生成最新的前后端代码。这种开发模式依然需要先创建数据模型，或者使用现有的数据模型，但不需要设计前端页面和后台服务，而是根据配置信息自动生成。

## 7、任讯应用开发平台的构成

为了方便大家快速开发企业级应用管理系统，任讯信息提供了一整套的应用开发解决方案，包括应用服务引擎、应用开发工具、工作流引擎、即时通讯工具和应用市场这五大部分，可以为多个领域的用户或合作伙伴提供支持，包括：

- **软件开发商和个人开发者**：通过采用低代码开发方式，可以大幅提升开发效率，任讯可以为其提供应用发布和推广渠道，提供授权控制和版本升级工具，提供协同开发管理支持。
- **软件经销商、服务商和 ICT 集成商**：通过学习并掌握低代码开发，可以增加软件研发业务，可以更好、更及时地为自身客户提供软件业务支持，可以研发行业性软件产品。
- **云计算厂商和设备生产商**：为主机增加企业级应用服务引擎、常用企业管理应用和开发工具，提升其产品竞争力，集成各种云资源和能力。
- **企业级客户**：通过学习并掌握低代码开发，企业可以自助开发各种简单业务应用，降低企业运营成本，快速响应公司发展和外部环境变化对于信息化管理的各种诉求。



TaskBuilder 属于整个任讯应用开发平台的一部分，除了 TaskBuilder，任讯应用开发平台还包括以下内容：

## 7.1 任擎 TasGINE 应用服务引擎

### 7.1.1 任擎简介

在开发一个企业级管理软件时，首先要考虑的是系统底层的基础性功能，例如组织结构管理、角色管理、操作权限管理、身份验证、系统日志记录和查询、系统 UI 界面、系统门户、消息推送、文档存储等等。

另外，还得考虑系统的稳定性、兼容性、安全性、可

扩展性和性能等，这些其实是每个软件系统必备的一些通用的基础功能或要求，如果所有这些都从头开发，需要耗费大量的时间和高昂的成本。

有些企业可能会找一些小公司或者个人开发软件，貌似可以用很低的成本开发一套软件，表面上看，该有的功能都有，但由于没有成熟稳定的底层框架，未来实际运行的过程中可能会



出现各种各样的问题。

另外，有很多企事业单位经过多年的信息化建设，内部开发或购买了多个信息化系统，但目前这些系统都是各自独立运行，没有互联互通，管理员需要挨个在各个系统内维护用户和权限信息，员工需要分别登录这些系统进行操作，不能统一进行身份认证和权限控制，各个系统的数据也没有互相共享，无法整合到一起进行查询统计，极大地影响了信息化建设的效果。

为了很好地解决上述问题，任讯信息从 2014 年年底开始设计和开发了一个企业级应用服务引擎--“任擎”。该引擎采用当前最先进的软件开发技术，结合我司十多年的信息化建设经验，将开发企业级应用所需的各种底层功能进行了合理的设计和精巧的封装，形成了丰富而强大的 API、功能组件和快速开发工具，基于该引擎，开发人员无须再浪费时间和精力去实现底层功能，也不用担心底层架构的稳定性、安全性以及性能等，仅需关注具体的业务功能开发即可，从而可以快速地开发出各种业务应用，大大降低系统开发成本。而且，该引擎通过智能化和便捷化的开发功能，大幅降低了开发企业管理类软件的难度，不需要专业的软件开发知识，只要有一定的计算机操作基础，经过简单的培训，即可开发各种管理应用，让企业在面对快速变化的市场和管理需求时，能以较低成本迅速响应。

经过十多年的潜心研发和持续完善，目前该引擎已实现功能丰富、操作简便、性能优良、安全稳定、界面美观、灵活可配、易于扩展的设计目标，并已经过众多客户的检验，包括央企、上市公司和省厅级政府单位，得到客户的一致好评。

## 7.1.2 任擎的核心功能

任擎由以下四个部分构成：

### 7.1.2.1 应用服务器套件

应用服务器套件为各项业务应用提供了基础的服务器端运行环境，包括应用服务器、文件服务器、消息服务器和网关服务器等多种服务器程序，支持分布式部署和弹性计算，可以根据

不同的负载提供最适合的部署方式,当用户量比较少时可以把所有服务器程序都安装在一台物理或虚拟主机内,也可以在有大量用户访问的情况下分别部署在多台主机内,而且应用服务器还可以根据各个应用的访问量进行拆分,可以为访问量非常大的应用单独部署应用服务器,从而可以满足海量用户的访问。

其中,应用服务器为主服务器,是运行 TaskBuilder 开发的应用所必需的服务器,我们平时所说的任擎服务器就是指应用服务器。

**说明:** 共享版仅包括应用服务器(在服务管理器内显示为任擎服务器)和文件服务器。

### 7.1.2.2 应用开发框架

为了方便程序员开发各种业务应用,我们基于自身十多年的信息化系统研发经验,将开发应用软件所需的大量基础性工作经过精心设计和潜心研发封装成了几十种开发组件和功能,程序员使用这些组件可以很容易地快速开发出各种业务应用。

应用开发框架将开发业务应用所需的各种基础功能进行了精心设计并封装成各种开发组件和 API,包括服务器端和客户端两个部分,服务器端主要提供服务路由、数据解析、数据访问、业务逻辑校验和控制、数据缓存、身份验证、会话管理等服务。客户端主要提供数据展示、数据绑定、控件参数解析、数据传输等服务,电脑端兼容 Windows、Linux 和 Mac OS 以及绝大多数 Web 浏览器,手机端兼容 Android 和 iOS。

### 7.1.2.3 应用集成接口

经过多年的信息化建设,很多单位内部已经部署了各种信息系统,例如财务、ERP、HR 和 CRM 等,这些系统一般都是由不同的软件厂商开发的,在技术架构、数据结构、功能设置等各个方面都可能不一样,各个系统目前都是各自为战,互不相连,数据没有共享,形成一个个信息孤岛。员工在办理各项业务时,需要挨个登录各个系统进行操作,不仅访问费劲,而且经常进行重复性劳动,浪费大量的人力物力,违背了信息化建设的初衷。

为了实现各个信息系统的整合，方便统一管理和用户访问，我们提供了丰富的集成接口和工具，包括组织结构集成接口、消息推送集成接口、文档存储集成接口、工作流程集成接口、业务数据集成接口、应用功能集成接口、信息门户集成接口和移动办公集成接口等，支持多种集成方式和通信协议，可以很方便地与第三方系统集成。

#### 7.1.2.4 后台管理系统

任擎内置一套强大的企业级后台管理系统，包括系统门户和系统维护两大部分，系统门户支持用户信息设置，支持多种样式风格，支持自定义功能菜单，支持设置系统 logo 和系统首页地址。系统维护提供了组织结构、系统群组、应用管理、计划任务、后台服务、系统日志、附件管理、数据字典、编码规则、系统设置和系统信息等丰富的功能，完全可以满足大部分企业管理系统的需求。

### 7.1.3 任擎优势

任擎已经将开发企业级应用和互联网平台所需的各种底层框架和基础功能都已实现，能够确保系统的性能、安全性、稳定性、兼容性和可扩展性等，开发人员只需关心具体的业务需求并实现即可，不用再花很多精力和时间去实现底层和基础功能，从而大大加快项目的开发周期，降低整体开发成本。

#### 7.1.3.1 分布式部署，弹性计算，按需扩展

任擎支持分布式部署和弹性计算，可以根据用户规模、访问频次灵活调整部署方式，当用户量比较少时，可以将所有服务器端程序都部署在一台主机上，当用户量比较多、访问量较大时，可以将整个平台按业务功能进行拆分，分别部署到不同的主机上，或者将访问量特别大的热点服务进行独立部署，突破单台服务器的硬件限制，且底层架构不需要重构，可以平滑迁移，避免对现有业务造成影响。

### 7.1.3.2 性能优良，无惧海量并发

任擎的设计目标不仅仅只是为了满足单个企业内部少量员工的访问，还可以用来构建能够承载海量用户并发访问的大型信息化系统。为此，任擎在底层采取了很多能够有效提升系统性能和承受海量并发访问的措施，包括前面说的分布式部署、弹性计算，以及负载均衡、事件驱动、热数据缓存、读写分离、非阻塞 I/O 等，能满足大型的信息化系统或互联网平台对性能和承载能力的苛刻要求。

### 7.1.3.3 多重保障，安全可靠

任擎提供了完备的安全保障体系，可以从硬件、网络、系统、应用和管理等多个层面为客户提供强大的安全保障，确保信息安全，避免信息泄露，防止数据损坏或丢失。

### 7.1.3.4 多终端、多系统全面兼容

为了兼容各种终端、操作系统和浏览器，任擎花费了大量的时间进行兼容性适配和测试，目前手机客户端支持 Android 和 iOS 系统，H5 页面支持嵌入到任讯、阿里钉钉和企业微信，电脑端支持 Google Chrome、Firefox、Edge、Safari、360、搜狗、QQ 等市面上主流的各种浏览器，开发人员不需要自己再花精力去适配。

另外，任擎已全面适配主流的国产软硬件厂商的产品，并已获得其兼容性认证，包括飞腾、统信、麒麟、人大金仓、武汉达梦等，且已经在省厅级单位的“信创”项目中成功应用。

### 7.1.3.5 支持多业务集成整合

任擎提供了丰富的集成接口和工具，包括组织结构集成接口、消息推送集成接口、文档存储集成接口、工作流程集成接口、业务数据集成接口、应用功能集成接口、信息门户集成接口和移动办公集成接口等，支持多种集成方式和通信协议，可以很方便地与第三方系统集成，可以将企业内部各种信息系统整合到一个统一的平台中。



## 7.2 任流 TaskFlow workflow 引擎

### 7.2.1 TaskFlow 简介

任流 TaskFlow 是本公司的第四代 workflow 引擎，从 2003 年开始研发第一个版本到现在已经有 20 多年历史，在这 20 多年的发展过程中，该产品不断迭代完善，已经非常成熟和稳定，并经过了上万家客户的检验。

使用 workflow 可以改进和优化业务流程，提高业务工作效率，实现更好的业务过程控制，是工作有序开展，规范执行，持续跟进的前提。TaskFlow workflow 引擎通过计算机技术的支持去定义、执行和管理 workflow，协调 workflow 执行过程中工作之间以及群体成员之间的信息交互。系统会自动记录工作办理过程中产生的各种数据，相关人员可以随时了解工作的办理进度，最后还能够对工作数据进行查询和统计。组织的所有流程性事务都可以通过 TaskFlow workflow 引擎进行申请、审批和查询统计，所有工作既可以在电脑上申请审批，在手机上通过移动客户端也可以申请审批。

TaskFlow workflow 引擎由以下功能构成：

### 7.2.2 表单设计器

使用表单设计器可以实现各项工作单据的在线设计，既可以使用图形化工具进行所见即所得的设计，也可以直接编辑 HTML 代码，实现复杂的单据格式。

TaskFlow 的表单设计器具有以下功能特点：

- 支持所见即所得的图形化编辑，支持文字、表格、图片、控件、链接、多媒体等丰富的内容格式；支持复制、粘贴、剪切、撤销、重做、缩进等常用编辑操作；支持调整字体、字号、字体颜色、背景色、对齐方式等。
- 支持从 Word 或 Excel 导入表单格式。

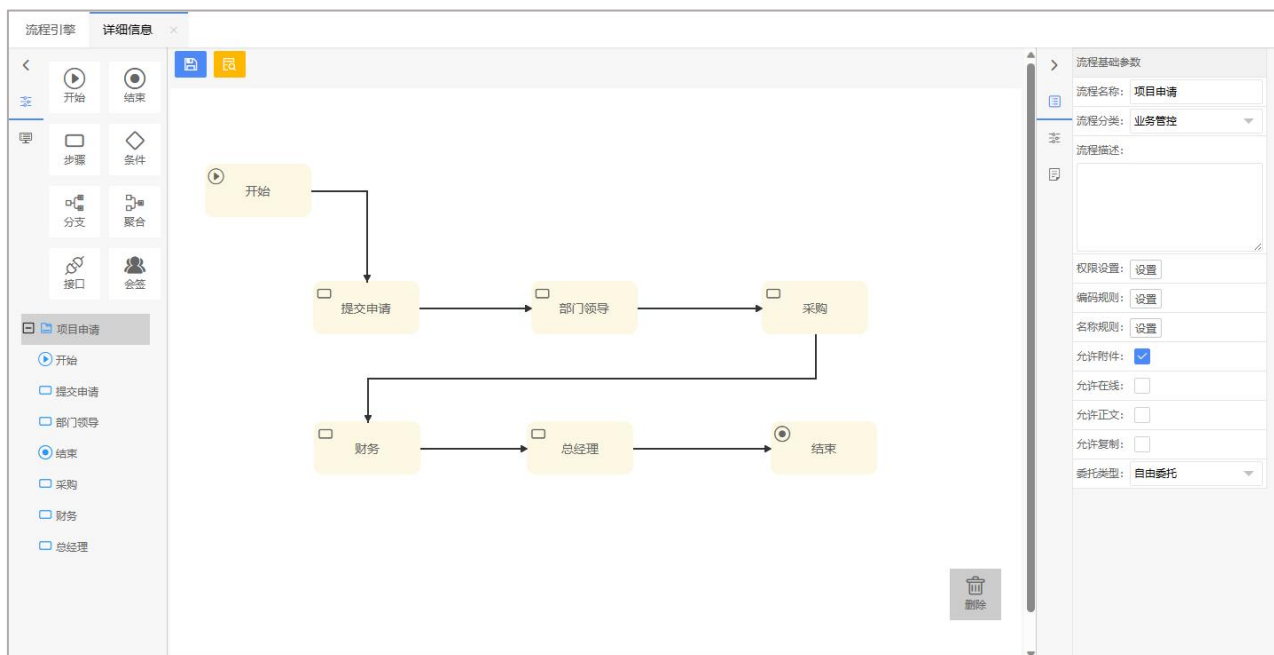
- 支持单行文本、多行文本、单选框、多选框、下拉列表、数值、日期、时间等常用控件，每个控件都可以设置字号、宽度、对齐方式、边距等详细参数。
- 支持工作编号、工作名称、用户姓名、部门名称、当前时间等宏变量。
- 支持多人会签，可以选择会签时需要显示哪些控件，包括用户名、时间、会签意见等。
- 支持电子签章，可以将手写签名扫描成图片后设置成电子签章，签章时输入密码即可显示。

The screenshot displays the '界面设计' (Interface Design) tab in TaskBuilder. The main canvas shows a form titled '合同审批单' (Contract Approval Form). The form is divided into several sections: '基本信息' (Basic Information), '合同信息' (Contract Information), and '产品明细' (Product Details). The '基本信息' section includes fields for '申请人' (Applicant), '所属部门' (Department), '岗位/职务' (Position/Post), and '申请日期' (Application Date). The '合同信息' section includes fields for '合同名称' (Contract Name), '合同编号' (Contract Number), '单位名称' (Unit Name), '联系人' (Contact), '合同类型' (Contract Type), '付款方式' (Payment Method), '合同金额' (Contract Amount), and '金额(大写)' (Amount in Large Characters). The '产品明细' section is a table with columns for '产品名称' (Product Name), '销售数量' (Sales Quantity), '单价' (Unit Price), and '金额' (Amount). The right-side panel shows the '属性' (Properties) of the selected control, including '控件编号' (Control ID), '控件类型' (Control Type), '宏类型' (Macro Type), '名称' (Name), and '隐藏' (Hidden).

- 支持动态表格，可以设置每一列要显示的内容，然后在填写时可以动态增加或删除。可以设置最小和最大行数。
- 支持为控件设置计算公式，当某个控件的值发生变化时，其他相关控件自动发生变化。
- 支持直接编辑表单的 HTML 代码。
- 支持表单脚本的编写，当表单发生某些事件时可以执行相关脚本实现某些复杂的功能。
- 支持表单 CSS 样式的统一编辑。
- 支持表单打印样式的设置，实现打印格式与填写、查看时的格式独立。
- 支持表单数据源设置，可以从其他流程的表单中选择数据，也可以通过接口从第三方系统中选择数据。

## 7.2.3 流程设计器

通过流程设计器可以实现各项工作的流程设置，包括流程基本信息设置、流程图设置、步骤设置、表单设置、附件模板设置、正文套头模板设置、监控权限设置、归档设置等。



TaskFlow 的流程设计器具有以下功能特点：

- 支持固定流和自由流，固定流必须事先设置好步骤，自由流则没有固定的步骤，而是在工作办理的过程中可以随意转交给任何一个人，直到有人选择结束。
- 一个流程可以支持多个表单，在办理时可以切换，以便分别查看和录入。
- 支持图形化的流程图设置，提供了开始、办理、判断、分支、聚合和结束等节点，可以拖

拽连接节点。可以随时添加或删除节点及连接线。

- 流程的每一步都提供了丰富的配置参数，完全可以满足各行各业客户的绝大多数的工作办理需求，包括主办人类型、经办人权限、办理人数限制、办理顺序、转交方式、转交后跳转方式、退回权限、退回方式、修改权限、回收权限、附件操作权限、正文编辑权限、是否必须签收、是否允许填写办理意见、是否允许评论、是否允许催办、是否允许修改工作名称、是否允许协办、是否允许传阅、是否允许告知、是否允许挂起、是否允许终止、是否允许归档、是否允许打印、是否允许导出等。
- 流程的每一步都可以精确地设置有权限办理的人员，可以任意选择、由指定的扩展页面选择、通过 SQL 语句选择或通过各种预置的条件组合确定人员范围。系统提供了上百种人员选择条件。另外，还可以设置必须经办人、固定主办人、排除在外的办理人、可以预先设置的人员等更丰富的选项。
- 每一步可以设置具体可以访问哪个表单里的哪些输入项，哪些输入项是必填项，哪些是对当前步骤不可见的。
- 可以为指定步骤设置转交条件，系统提供了一个智能表达式设计器，将常用的运算符、判断符、条件符、函数、宏变量和表单输入项全部列出，用户不需要专业的计算机知识，点点鼠标就能设计出复杂的条件判断表达式。
- 每一步都支持办理时限设置，提供了多种计时方式和时长供选择，超时后可以发送提醒信息。
- 转交工作时默认只会提醒下一步，如果还需提醒其他步骤的人或在进行其他操作时也需要提醒，那么就可以手动设置提醒，系统支持转交提醒、退回提醒、催办提醒、委托提醒、传阅提醒、告知提醒、移交提醒等多种提醒类型，每一步可以添加多个提醒，每个提醒都可以设置接收人和提醒内容格式。

- 支持步骤扩展设置，当某一步在进行转交、退回、接收和收回等操作时，可以设置需要执行的扩展程序。可以选择我们为第三方 IT 厂商提供的系统对接扩展，例如在本平台内填写的采购申请在审批通过后，可以通过相应的扩展接口连接金蝶或用友的 ERP 系统自动生成该系统内的采购单。

## 7.2.4 工作申请

在表单和流程设置好后，用户就可以在流程中心选择相关的流程创建申请，填写申请内容，然后按照设置好的步骤一步一步转交和审批了。如果流程比较多，可以对流程进行分类，以便按类别查找。每个类别和每个流程都可以设置访问权限，用户只能看到有权使用的流程。创建某个流程的工作申请时，工作编号、工作名称等基本信息可以手动输入，也可以自动生成。

## 7.2.5 工作审批

工作转交后，下一步的办理人会收到相关提醒信息，支持手机 APP、网页弹窗、短信等多种形式的提醒，点击提醒可以直接打开工作审批页面。在系统首页的待办工作列表或者流程中心的待办工作里也会看到该工作。打开该工作后即可查看该工作的详细信息，如果不同意可以退回，如果没问题就填写审批意见，然后继续一步步往下转交，直到结束。另外，根据管理员设置的权限，办理人还可以进行填写附加意见、上传或查看附件、传阅、打印、导出等操作。流程提交后可以追回，审批可选择同意、不同意、退回上一级。

## 7.2.6 工作监控

为了方便单位领导或各项业务的主管部门主管了解相关工作的办理进度，管理员可以为这些领导设置相关流程的监控权限，这些领导就可以随时监控这些流程的办理情况，在出现问题或卡壳时能及时处理。监控权限分为查看、移交和代办三个级别，查看权限仅能看不能操作；移交是可以将当前办理权限授权给其他人；代办是指可以代替当前步骤的办理人进行办理。

## 7.3 任讯 TaskMsg 即时通讯工具

### 7.3.1 任讯简介

任讯是专为企业内部管理而设计的一款办公专用客户端软件，可以作为企业办公的入口，包括手机端和电脑端两种客户端，内置消息提醒、通讯录、即时通讯等基础功能。另外，还可以在工作台内随时增加扩展功能。

任讯手机客户端支持 iOS 和 Android 系统，集成了手机系统的消息推送、拍照、定位、扫码等丰富的功能，可以用于各种业务工作的提醒、数据填报和工作审批等场合。

### 7.3.2 任讯核心功能

#### 7.3.2.1 消息中心

消息中心用来集中显示通过任讯收到的各类信息。可对消息进行分类，包括任务的消息、聊天的消息等。聊天消息又分为单聊和群聊，每种不同类型的消息分行显示，点开某一类消息即可看到详细内容。

- 支持未读消息突出显示
- 支持未读消息数量显示
- 支持同类消息分组显示
- 支持消息类别图标设置
- 支持关键词模糊查找
- 支持消息置顶显示
- 支持删除消息记录



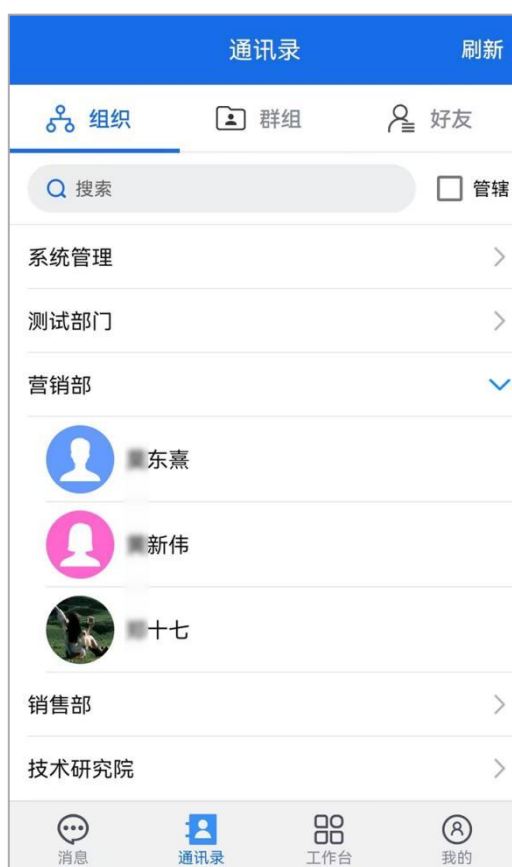
### 7.3.2.2 通讯录

为了方便用户查找同事或好友的联系方式，任讯内置强大的通讯录功能，包括组织内部通讯录、外部好友通讯录和群组通讯录三个部分。其中，内部通讯录由管理员统一维护，不需要用户自己手动添加，外部好友由用户自己添加和维护。在组织内部通讯录和外部好友通讯录内找到联系人后，可以查看联系人的详细信息，可以直接给联系人打电话、发邮件，或者发起聊天或任讯。在群组内找到相应群组后，即可进入群聊窗口进行聊天。

- 支持多层组织结构，不限层次，可以层层展开
- 支持大用户量的组织，即使上万人也不会出现卡顿
- 支持显示格式个性化设置，联系人列表内可以分别设置左侧、右侧、下部具体显示什么内容，可选的内容包括姓名、职务、手机号、集团小号、邮箱、个性签名等
- 支持查看权限设置，可以设置指定的群组或人员在通讯录内具体可以查看哪些部门或

人员，包括可以查看全部人员、可以查看本单位人员、可以查看本部门人员、可以查看领导、可以查看下属等选项

- 支持被查看权限设置，可以设置指定的人员的信息在通讯录内只可以被哪些群组或人员看到
- 支持人员显示顺序设置，可以为每个人设置在通讯录内显示的序号
- 支持人员模糊查找，输入关键字后能自动列出符合条件的联系人
- 支持一人多职，即一个人可以同时多个部门任职
- 群组支持系统群组、部门群组、用户群组和临时讨论组，系统群组由管理员统一维护，普通用户不能创建，也不能加人或踢人；部门群组对应当前用户所在的部门，成员为该部门的所有人员；用户群组为当前用户创建或加入并收藏的群组；临时讨论组为当前用户创建或加入但未收藏的群组





### 7.3.2.3 即时通讯

任讯内置标准的即时通讯功能，能实现组织内部的畅捷沟通，并且所有沟通记录在服务器端都会自动备份，管理员可以查询和导出，一旦出问题可以查证，即使员工离职也不会导致沟通记录丢失，做到安全可控。

- 支持单聊和群聊，群聊人数无限制
- 支持文字、表情、图片、文件、语音、小视频、位置等格式的内容
- 支持直接拍照发送图片
- 支持客户端本地记录的分类和模糊查找，支持本地聊天记录删除
- 支持实时语音和视频通话，最多可以支持 9 人同时实时通话
- 群聊支持加人、删人、退出、解散等操作，可以设置群名称，可以设置消息免打扰
- 群聊时，支持@符号选人，不管对方是否设置消息免打扰，都会提示对方



### 7.3.2.4 工作台

除了上述基础功能外，任讯还支持通过工作台嵌入手机端 H5 应用，可以随时按需添加。



## 7.3.3 任讯优势

### 7.3.3.1 支持按需定制，满足个性化需求

可以按照客户的个性化需求进行定制化开发，从而让任讯更加契合客户的管理需求。

### 7.3.3.2 提供客户端扩展，实现复杂功能

提供客户端扩展 API，可以轻松实现拍照、定位、录音、扫码等扩展功能。

### 7.3.3.3 可以私有化部署，系统安全可控

针对比较大的企业或政府机关，如果有自己的机房和专业的信息化维护队伍，可以将任讯消息服务器安装在客户自己的服务器上，以便满足客户更高的安全要求。

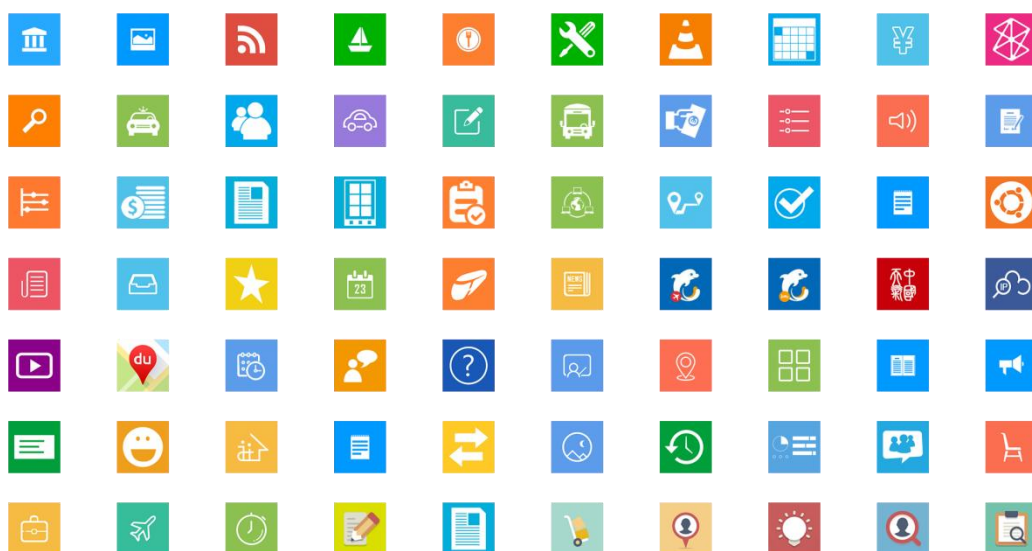
### 7.3.3.4 提供应用市场，提供海量应用

可以从任讯企业应用市场安装各种应用，为各行各业的客户提供现成的行业应用。

## 7.4 任集 TaskStore 应用市场

为了方便 TaskBuilder 开发者将自己开发的应用分享给更多的人使用，也为了方便大家查找和安装别的开发者已经开发好的应用，我们建设了企业级的应用市场--任集 TaskStore，TaskStore 为企业级用户提供了丰富的日常管理应用和行业应用，可以满足不同行业客户的各种管理需求，TaskStore 也为软件开发者提供了应用发布和销售的渠道，可以共享平台的基础服务能力和海量的客户资源。

另外，TaskBuilder 开发者之间也可以通过应用市场分享自己制作的功能模版、扩展组件等，形成互利互助的知识共享平台。



## 二、TaskBuilder 开发平台安装配置

### 1、任擎服务器 Windows 版安装配置

用 TaskBuilder 开发的应用，必须先安装和配置好 任擎 服务器，因为要通过 任擎 解析和编译 tfp 页面，而且， 任擎 也可以作为 Web 服务器和后台应用服务器，负责接收和响应静态文件请求（html、css、前端 js 和图片等），解析和执行后台服务代码，并提供身份验证、访问权限校验、会话管理、文件上传下载、数据缓存管理、计划任务管理、日志记录和跟踪、数据库连接监管等众多基础服务。

所以在使用 TaskBuilder 进行应用开发前，需要先安装和配置好 任擎 服务器。下面是安装和配置 任擎 服务器的详细操作说明：

#### 第一步：下载任擎服务器

点击下面的链接下载 任擎 服务器最新版本的绿色安装包：

<http://taskbuilder.oss-cn-beijing.aliyuncs.com/tasgine.zip>

将下载的 任擎 服务器解压之后即可使用。

**注意：** 请不要将任擎服务器安装文件直接解压在电脑桌面上，建议解压到电脑的非系统盘上，例如 D 盘或 E 盘，因为本安装包为绿色安装，解压后的文件就是系统实际的运行文件，如果解压到电脑桌面上，一是占用系统盘空间，二是可能会被不小心删掉。

#### 第二步：使用安装向导配置任擎服务器

##### (1) 解压运行“安装配置向导”

解压 任擎 安装包 tasgine.zip，双击运行解压后的 tasgine 文件夹中的 安装配置向导.exe。

|                                 |                  |                |           |
|---------------------------------|------------------|----------------|-----------|
| ServerManager.exe               | 2022/6/21 11:48  | 应用程序           | 49 KB     |
| ServerManager.xml               | 2022/12/27 13:42 | XML 文档         | 1 KB      |
| start.bat                       | 2021/1/8 16:01   | Windows 批处理... | 1 KB      |
| start.js                        | 2022/1/9 21:53   | JS 文件          | 2 KB      |
| startTcpFileServer.js           | 2022/12/27 13:42 | JS 文件          | 1 KB      |
| tasgine.exe                     | 2023/1/7 16:52   | 应用程序           | 73,097 KB |
| TaskMsgService.exe              | 2020/7/3 13:24   | 应用程序           | 11 KB     |
| uninstall_service_dotnet2.0.bat | 2022/11/30 23:13 | Windows 批处理... | 1 KB      |
| uninstall_service_dotnet4.0.bat | 2022/11/30 23:13 | Windows 批处理... | 1 KB      |
| 安装配置向导.exe                      | 2022/7/27 19:48  | 应用程序           | 128 KB    |

## (2) 设置数据连接参数

打开安装向导之后，首先要设置数据库有关的参数，如下图所示：

任擎安装配置向导

**任擎 TASGINE**  
企业级应用服务引擎

Based on  
 

**安装步骤**

- 设置数据库参数
- 设置任擎参数
- 执行安装配置
- 配置完成

**设置数据库参数**

任擎(Tasgine)服务器运行时必须使用数据库，在使用本向导之前，请先安装好数据库管理软件，然后再使用本向导连接数据库创建任擎服务器所需要的表和相关视图，并导入初始数据。数据库和任擎可以在不同的电脑上。

|        |           |     |         |
|--------|-----------|-----|---------|
| 数据库类型: | MySQL     | 名称: | tasgine |
| 数据库地址: | 127.0.0.1 | 端口: | 3306    |
| 用户名:   | root      | 密码: |         |

如果数据库已存在，则会删除任擎服务器有关的表和视图，然后重建，请提前做好备份。  
如果数据库和任擎不在同一台电脑，请确保数据库所在电脑的防火墙已开放数据库端口。  
如果数据库在云服务器上，除了防火墙，在云服务器的安全组里也需要开放数据库端口。

[MySQL数据库安装操作指南](#)

下一步 取消

www.taskmsg.com  
技术支持QQ群: 870548663

数据连接参数包括以下几个参数：

- **数据库类型：**目前可以选择 MySQL 或者 MS SQLServer，如果您还没有安装任何数据库软件，需要先安装，推荐安装 MySQL，具体参考：MySQL 数据库安装操作指南。
- **数据库名称：**数据库名称默认为 tasgine，可以修改为其他名称，如果数据库已存在，会直接在该数据库中创建任擎所需要的系统表和视图，如果这些表或视图已经存在，会删除



重建。所以，为了避免数据丢失，请在运行本向导之前备份好现有的数据库。

- **数据库地址：**数据库软件所在电脑的 IP 地址，默认为：127.0.0.1，表示当前电脑，如果引擎和数据库在同一台电脑上，则此处不用动，如果不在一台电脑上，则需要设置数据库所在电脑的 IP 地址。
  - **端口：**数据库软件的网络端口，MySQL 数据库默认的端口为 3306，MS SQLServer 数据库默认的端口是 1433，如果数据库不是用的默认端口，则需要在此修改。
  - **用户名：**数据库软件的登录账号，MySQL 数据库默认的管理员账号为 root，MS SQLServer 数据库默认的管理员账号为 sa，如果要用其他账号登录，则在此修改配置。
  - **密码：**上面数据库登录账号的密码。
- 设置完之后，点击下一步按钮。

### (3) 设置引擎服务器参数

在此设置引擎服务器有关的参数，如下图所示：

引擎安装配置向导

**任 擎 TASGINE** 企业级应用服务引擎 Based on node.js

#### 设置引擎参数

引擎(TasGINE)服务器提供了丰富的配置项，本向导仅配置几项基本参数，更多参数可以在引擎服务器所在目录的/config/TasGINE.json文件中进行配置。

引擎服务器端口: 8001 请确认端口未被其他软件占用

admin 账号密码: 请输入密码 admin是引擎后台管理员账号

新建用户默认密码: 请输入密码 在后台新建用户时使用该密码

☐ 安装为Windows服务 (开发环境不建议安装)

[引擎服务器配置参数详细说明](#)

www.taskmsg.com 技术支持QQ群: 870548663

上一步 下一步 取消

引擎服务器参数说明：

- **引擎服务器端口：**默认为 8001，可以修改为其他端口，但要确保该网络端口没有被别的

程序占用。

- **admin 账号密码**：指用 admin 账号登录任擎后台管理系统时的密码，请务必记住此处设置的密码，后续登录任擎后台管理系统或在 TaskBuilder 里连接任擎服务器时，都需要用到该密码。将来如果要修改 admin 的密码，可以用 admin 账号登录后台管理系统，然后把鼠标放到系统右上角的用户名或头像上，在弹出的下拉菜单里，点击修改密码，即可打开密码修改窗口进行修改。
- **新建用户默认密码**：指在任擎后台管理系统内新建用户时设置的默认密码。将来如果要修改该默认密码，可以用 admin 账号登录后台管理系统，然后进入系统维护的系统设置模块，在基本参数里，即可进行修改。
- **安装为 Windows 服务**：如果需要在电脑开机启动时自动运行任擎，则需要勾选该选项，否则不需要选中。

设置完之后，点击下一步按钮。

## (4) 开始安装配置

点击 开始配置 按钮即可开始执行安装配置。



当显示配置成功后点击下一步。



最后, 可以直接点击 **完成** 按钮完成设置, 也可以先选择 **打开任擎服务器管理器**, 然后再点击完成按钮, 这样会自动打开 **任擎服务器管理器**, 在该软件内启动或停止任擎服务器。上面代码里的 123456 是 MySQL 数据库 root 账号的密码, 请修改成实际的密码。

## (5) 将任擎服务器安装为 Windows 的系统服务

如果想让任擎服务器随 Windows 系统启动后自动运行, 可以运行任擎服务器目录下的以下批处理命令, 将任擎服务器安装为 Windows 系统服务:

**install\_service\_dotnet2.0.bat** (Windows 系统里的 .net 运行时为 4.0 以下版本)

或

**install\_service\_dotnet4.0.bat** (Windows 系统里的 .net 运行时为 4.0 版本)

如果要卸载该服务, 则运行对应的卸载批处理程序即可:

**uninstall\_service\_dotnet[2.0 或 4.0].bat**



## (6) 常见问题

1、如果网络端口已经被别的软件占用，启动任擎服务器时，会在服务管理器内显示类似以下的错误：

```
Error: listen EADDRINUSE: address already in use :::8001
```

2、如果 MySQL 数据库的账号或密码没有配置正确，启动任擎服务器时，会在服务管理器内显示类似以下的错误：

```
ER_ACCESS_DENIED_ERROR: Access denied for user 'aaa'@'localhost'
```

## 2、任擎服务器 CentOS 版安装配置指南

下面是安装和配置 任擎 服务器 centos 版的详细操作说明：

### 第一步：下载并初始化任擎服务器

点击下面的链接下载 任擎 服务器 centos 版本最新的绿色安装包：

<http://taskbuilder.oss-cn-beijing.aliyuncs.com/tasgine.tar.gz>

自动配置脚本当前仅支持指定目录下运行，所以第一步必须先创建/home/taskmsg 目录。

```
mkdir /home/taskmsg
```

```
cd /home/taskmsg
```

将下载的 任擎 服务器压缩包放到/home/taskmsg/目录中并解压。

```
tar -zxvf tasgine.tar.gz
```

解压上述文件后，会在/home/taskmsg/目录里创建一个名为 tasgine 的目录，需要进入该目录，才能执行后续操作。

请不要将压缩包里 tasgine 目录里的文件直接解压到/home/taskmsg/目录

### 第二步：配置任擎服务器

进入任擎 任擎 (Tasgine) 目录：

```
cd /home/taskmsg/tasgine
```

执行自动配置脚本（缺确保本机已经安装好 mysql 数据库相应版本），将自动完成开机启动自启动配置、数据库创建和导入、防火墙端口配置、任擎配置文件修改，并启动 任擎 (Tasgine) 服务器。

```
sh init.sh 数据库用户名 数据库密码
```

```
开始初始化任擎+++++
正在配置防火墙+++++
Warning: ALREADY_ENABLED: 8001:tcp
success
success
配置防火墙完成-----

正在安装任擎服务+++++
任擎服务安装完成-----

正在初始化数据库+++++
Warning: Using a password on the command line interface can be insecure.
数据库初始化完成-----

正在初始化配置文件+++++
配置文件初始化完成-----

tasgine正在启动服务.....
pid:3438
tasgine启动成功
任擎初始化完成-----
```

如果一切顺利，你可以开始体验虐 Tasgine 的愉悦了。

## 常用命令

查看任擎状态：

```
service tasgineServer list
```

启动任擎：

```
service tasgineServer start
```

停止任擎：

```
service tasgineServer stop
```

重启任擎：

```
service tasgineServer reload
```

### 3、任擎服务器 macOS 版安装配置指南

下面是安装和配置 任擎 服务器 macOS 版的详细操作说明：

#### 第一步：下载并初始化任擎服务器

点击下面的链接下载 任擎 服务器 macOS 版本最新的绿色安装包：

[http://taskbuilder.oss-cn-beijing.aliyuncs.com/tasgine\\_mac\\_arm.zip](http://taskbuilder.oss-cn-beijing.aliyuncs.com/tasgine_mac_arm.zip)

**说明：**当前的任擎服务器 Mac 版仅支持运行在搭载 Apple Silincon 处理器的 ARM 版 macOS 系统上，暂不支持 Intel 芯片的 macOS 系统。

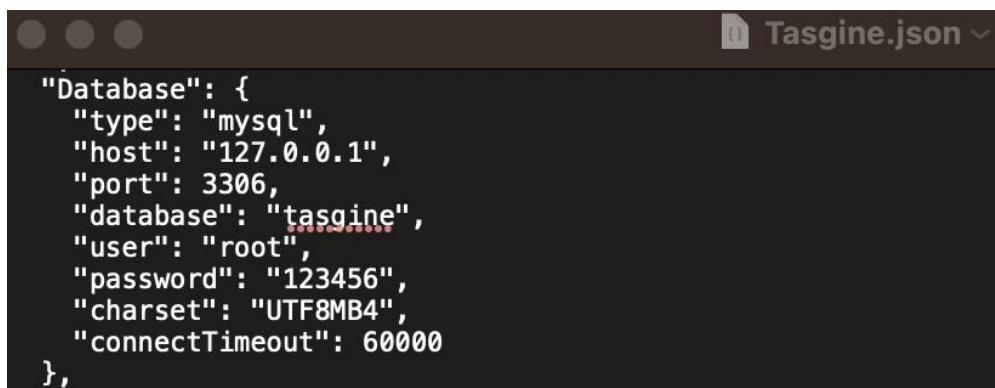
将该压缩包解压到指定位置，例如：/Users/用户名/Desktop/。解压上述文件后，将自动生成一个名为 tasgine\_mac 的目录，需要进入该目录，才能执行后续操作。

#### 第二步：配置任擎服务器

1、进入 tasgine\_mac 目录下的 config 子目录，使用文本编辑器打开该目录下的 Tasgine.json 文件：



2、修改该文件中对应的 MySQL 数据库的账号和密码，然后保存：



**特别提示：**此步骤最为关键，正确设置路径与否将直接关系到后续任擎服务器能否正常启动。

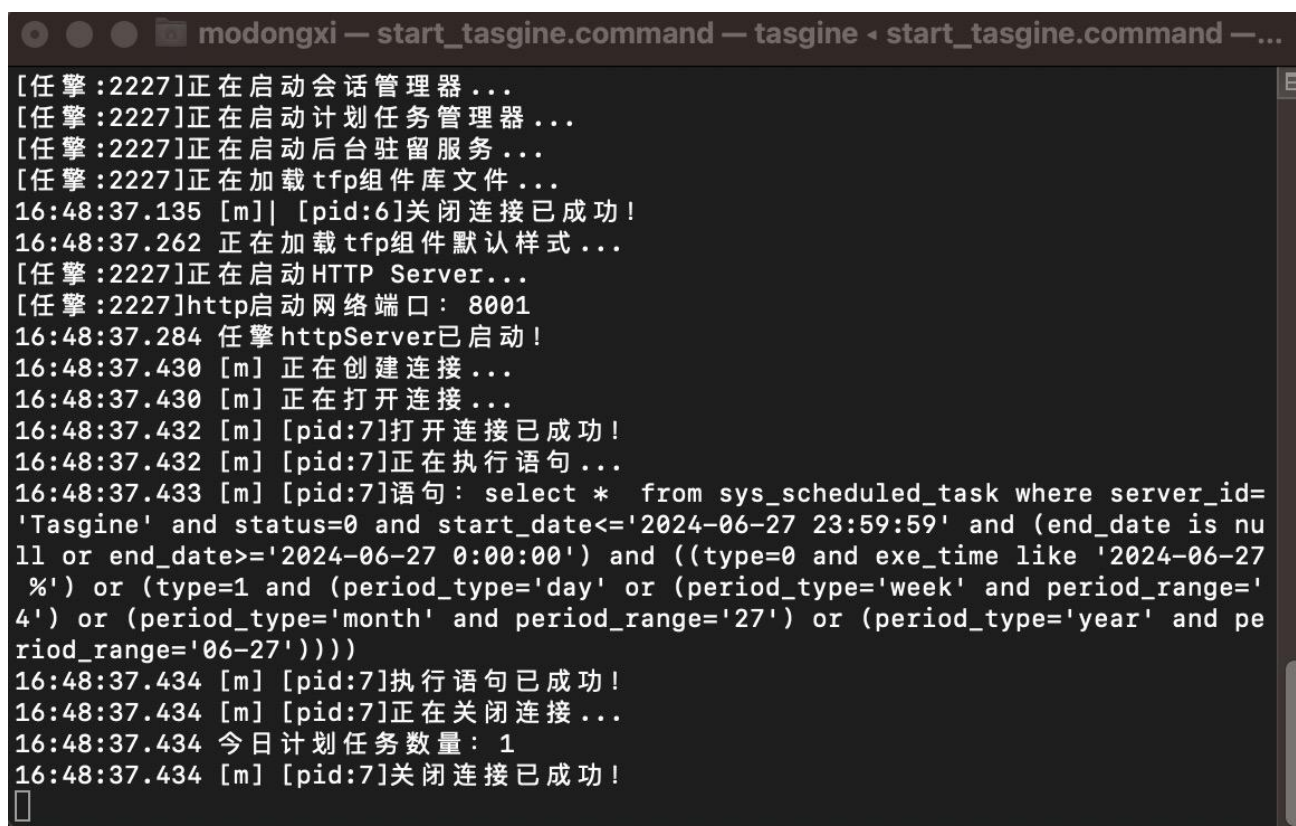
2、返回 `tasgine_mac` 目录, 用文本编辑器打开该目录下的 `start_tasgine.command` 文件, 确认里面第一行的路径是否是当前目录的路径, 默认为 `Desktop/tasgine_mac`, 如果不是, 则需要修改, 然后保存:



```
start_tasgine.command
cd Desktop/tasgine_mac
./tasgine ./start.js
```

## 第三步：启动任擎服务器

双击 `tasgine_mac` 目录里的 `start_tasgine.command` 文件 (请确保本机已经安装好 `mysql` 数据库并已经创建好 `tasgine` 数据库文件), 就可以启动任擎服务器。



```
modongxi — start_tasgine.command — tasgine < start_tasgine.command —...
[任擎:2227]正在启动会话管理器...
[任擎:2227]正在启动计划任务管理器...
[任擎:2227]正在启动后台驻留服务...
[任擎:2227]正在加载tfp组件库文件...
16:48:37.135 [m] [pid:6]关闭连接已成功!
16:48:37.262 正在加载tfp组件默认样式...
[任擎:2227]正在启动HTTP Server...
[任擎:2227]http启动网络端口: 8001
16:48:37.284 任擎httpServer已启动!
16:48:37.430 [m] 正在创建连接...
16:48:37.430 [m] 正在打开连接...
16:48:37.432 [m] [pid:7]打开连接已成功!
16:48:37.432 [m] [pid:7]正在执行语句...
16:48:37.433 [m] [pid:7]语句: select * from sys_scheduled_task where server_id=
'Tasgine' and status=0 and start_date<='2024-06-27 23:59:59' and (end_date is nu
ll or end_date>='2024-06-27 0:00:00') and ((type=0 and exe_time like '2024-06-27
%') or (type=1 and (period_type='day' or (period_type='week' and period_range='
4') or (period_type='month' and period_range='27') or (period_type='year' and pe
riod_range='06-27'))))
16:48:37.434 [m] [pid:7]执行语句已成功!
16:48:37.434 [m] [pid:7]正在关闭连接...
16:48:37.434 今日计划任务数量: 1
16:48:37.434 [m] [pid:7]关闭连接已成功!
```

如果看到上述提示, 表示任擎配置正确, 已成功启动!

## 停止任擎

在命令行窗口按下 `Control+C` 即可关闭任擎服务器; 或者直接关闭命令行窗口亦可。

## 4、任擎服务器 Docker 版安装配置指南

### 第一步：准备宿主机运行环境

在安装和运行任擎服务器 Docker 容器之前，需要先在宿主机中创建好相关目录，以便存储任擎服务器的配置文件、附件、日志和数据库文件等信息。

在宿主机中执行以下命令创建相应目录：

```
mkdir -p /home/tasgine
mkdir -p /home/tasgine/config
mkdir -p /home/tasgine/files
mkdir -p /home/tasgine/logs
mkdir -p /home/tasgine/mysql_data
```

### 第二步：安装配置任擎版 MySQL 数据库容器

为了方便开发者，我们制作了一个 MySQL 数据库的 Docker 镜像，里面已经包含了运行任擎服务器所必需的表、视图、函数和初始数据，安装完该镜像，启动该容器后即可直接使用。

从 Docker 仓库下载任擎版 MySQL 镜像文件的命令如下：

```
docker pull taskmsg/mysql-tg:1.0.1
```

启动任擎版 MySQL 容器的命令如下：

```
docker run -p 3306:3306 --name mysql-tg -v
/home/tasgine/mysql_data:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=123456
taskmsg/mysql-tg:1.0.1
```

运行以上命令后，会启动任擎版 MySQL 容器，网络端口为 3306，root 账号的密码为 123456，数据库文件会存储在宿主机的 /home/tasgine/mysql\_data 文件夹中。

上面命令中的网络端口和 root 账号的密码可以根据需求自行修改。

执行以下命令可以测试任擎版 MySQL 容器是否启动成功：

```
docker exec -it mysql-tg /bin/bash;
mysql -h127.0.0.1 -P3306 -uroot -p123456;
```

```
use tasgine;  
select * from sys_org;
```

这里的网络端口和 root 账号密码要和第三步设置的一致。如果可以正常看到返回数据，则表示任擎版 MySQL 容器已启动成功。

## 第三步：设置任擎服务器配置文件

从 TaskBuilder 官网下载任擎服务器安装包（Windows 版或 Linux 版都可以），将安装包里的 tasgine\config 目录里的文件解压或上传到第一步创建的 /home/tasgine/config 目录中。

如果第二步启动 MySQL 容器时，网络端口或 root 账号的密码不是用的默认的，则需要用文本编辑器打开 config 目录中的 Tasgine.json 文件，修改 Database 设置项里的数据库连接参数，这些参数的作用说明如下：

host: 数据库服务器地址

database: 数据库名称

user: 数据库用户名

password: 密码

schema: 跟 database 参数的值要一致

## 第四步：安装配置任擎服务器容器

运行以下命令可以下载安装任擎服务器镜像：

```
docker pull taskmsg/tasgine:4.3.4.240101
```

运行以下命令可以启动任擎服务器容器：

```
docker run -p 8001:8001 --name tasgine -v
```

```
/home/tasgine/config/Tasgine.json:/tasgine/config/Tasgine.json -v
```

```
/home/tasgine/logs:/tasgine/logs -v /home/tasgine/files:/tasgine/files
```

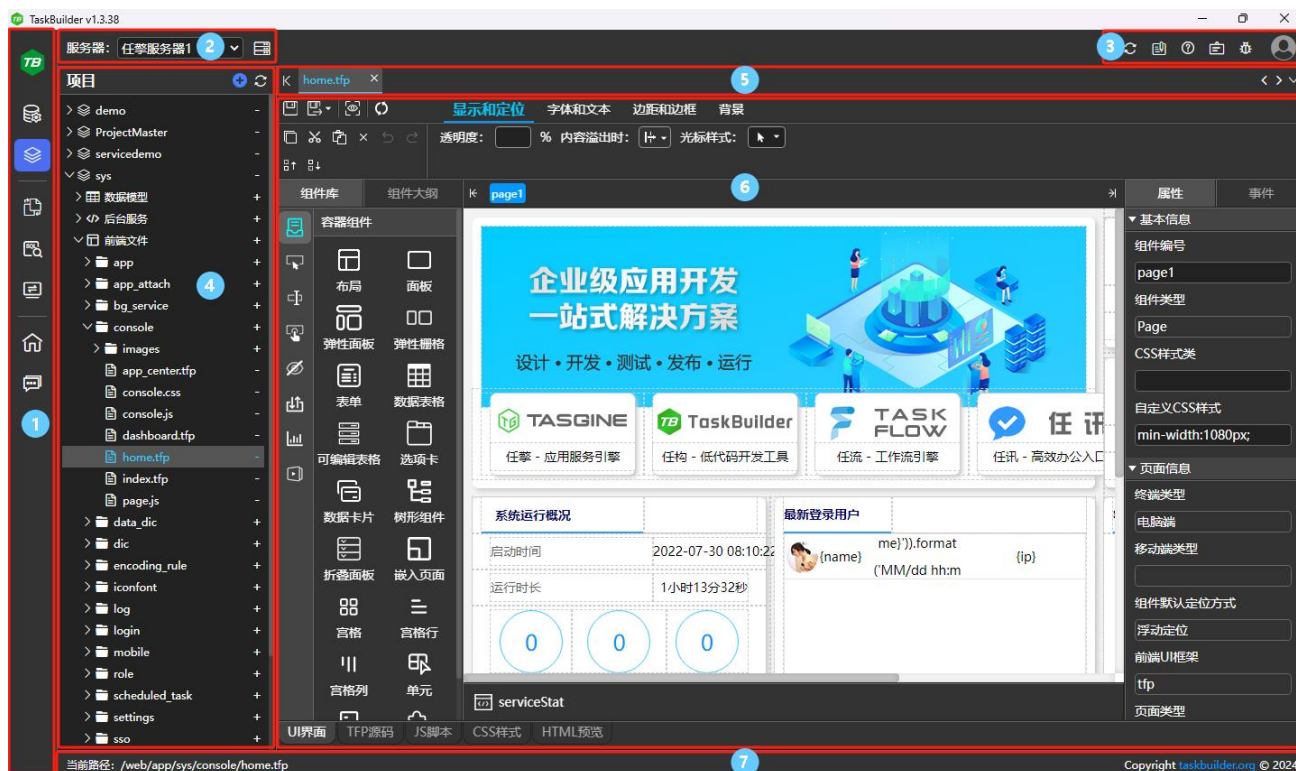
```
taskmsg/tasgine
```



## 三、TaskBuilder 开发环境简介

### 1、TaskBuilder 主界面

TaskBuilder 的主界面分为如下图所示的 7 个区域：



这 7 个区域的作用简要介绍如下：

1、**系统侧边栏**：在整个界面的左侧显示了 TaskBuilder 的 Logo 图标和各种系统功能，包括

数据源设置 、项目资源管理 、文件同步工具 、SQL 执行工具 、后台服务测试工具 、登录后台管理首页 、进入社区  等；其中，点击数据源设置和项目资源管理这两项的图标后，区域 4 内的内容会随之发生变化。

2、**服务器设置**：在此查看和设置任擎服务器的信息。应用系统的代码都是保存在任擎服务器上的，TaskBuilder 必须连接任擎服务器才能进行相关操作，且同一时间只能连接一个任擎服务器，默认连接服务器列表中的第一个服务器，可以打开服务器列表选择其他服务器进行切换，切换服务器后，区域 4 内的内容会随之发生变化。

- 3、**辅助工具栏**：在主界面右上角目前提供了 6 个图标，可以实现：刷新整个系统界面、查看和反馈意见、查看在线帮助、查看升级公告、显示调试工具、查看用户信息；
- 4、**系统资源列表**：默认显示项目资源列表，点击系统菜单里的数据源按钮，会显示当前任擎服务器上已配置的数据源信息；
- 5、**系统标签栏**：以标签的形式显示当前已打开的所有功能页面的标题，点击某个标签就会在区域 6 内显示这个功能页面对应的界面和内容；
- 6、**功能操作区**：用来显示 TaskBuilder 各项功能的操作界面，包括文件管理器、SQL 编辑器、数据模型设计器、后台服务设计器、前端页面设计器等。
- 7、**系统状态栏**：用来显示一些提示信息，包括当前打开的文件路径、错误提示等。

下面详细介绍一下各个区域内的具体功能。

## 2、系统侧边栏

在 TaskBuilder 主界面左侧的系统侧边栏中提供了 7 个按钮，可以实现以下功能：

### 2.1 数据源管理

任擎服务器默认必须要连接一个数据库作为系统数据库，在该数据库中需要存放组织结构、系统群组、应用信息、访问权限等运行任擎所必须的一些基础数据，开发者可以直接使用该数据库进行各种应用的开发。

目前任擎安装包仅提供了 MySQL 和 MS SQL Server 这两种数据库的初始化语句（在安装包的 tasgine\db 目录内），如果开发者还想访问其他类型的数据库，或者现有信息系统的数据库，则可以使用 TaskBuilder 里的数据源功能来实现。

在系统侧边栏中点击数据源管理图标后，会在系统资源列表内显示当前任擎服务器上配置的所有数据源信息，可以在此添加、查看、修改和删除数据源，其中，任擎默认数据库是运行任擎服务器所必须的，不能删除。





在此可以配置任意多个外部数据库的连接信息，支持 MySQL、MS SQL Server、PostgreSQL、Vastbase、人大金仓等多种数据库。在后续创建数据模型时，就可以选择从指定数据源（数据库）导入表或视图，或者将新建的数据模型保存到指定数据源。然后在后台服务里，对这个数据模型进行增删改查操作时，就会自动连接这个数据源对应的数据库进行相关操作。

点击数据源列表顶部工具栏里的蓝色加号按钮，可以打开 **新建数据源** 对话框，如下图所示：

在此可以配置数据源的相关信息，其中 ID 和名称可以任意设置，只要与现有数据源的 ID 或名称不重复即可，其他参数需要按照准备连接的数据库的相关要求进行配置。填写完连接参

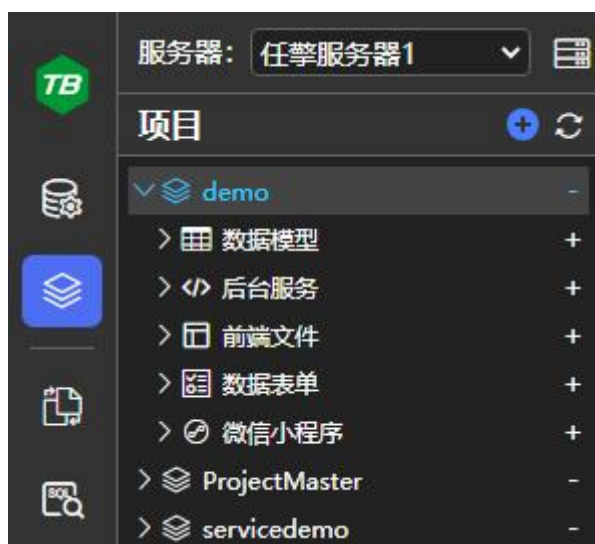
数后，可以点“测试”按钮进行连接测试，如果参数设置有误，会显示连接失败的错误。

在数据源列表内，点击现有数据源的名称，可以打开 **修改数据源** 对话框，可以在此修改现有数据源的相关参数，点确定进行保存。

在数据源列表内，点击某个数据源右侧的减号，可以删除该数据源。

## 2.2 项目资源管理

点击系统侧边栏里的项目图标 ，会在系统资源列表里显示当前任擎服务器上所有项目的各种资源列表，包括数据模型、后台服务、前端文件、数据表单和微信小程序等。



项目资源管理器用来对开发者自己开发的软件项目进行管理，这里的“项目”是指仅供开发者自己或开发者指定的特定用户使用、不需要发布到任讯应用市场让公开市场上的其他客户下载安装和使用的应用系统。

点击项目资源管理器顶部工具栏上的蓝色加号按钮，可以打开新建项目的对话框，如下图：



输入项目代码和说明后，点击确定按钮，即可创建一个新的项目。

在任擎安装包中默认提供了以下几个项目：

- **sys**：该项目为任擎服务器内置的后台管理系统，不能删除，如果要个性化修改该项目中的相关文件，请在修改后进行备份，否则后续升级可能会被覆盖。
- **demo**：该项目是我们为开发者提供的各种功能的演示示例，供开发者参考。
- **ProjectMaster**：该项目为一个简单的进销存系统的示例，目的是为了展示 TaskBuilder 的相关功能，并不是完全按照完善的进销存管理系统的要求进行开发的，功能非常粗糙，还有很多细节不完善，所以仅供参考，不要用于实际的业务管理。

在项目资源列表里，各种资源采取树形结构显示，可以层层展开查看，项目为一级节点，在每个项目下面，包含以下几个子节点：

### 2.2.1 数据模型

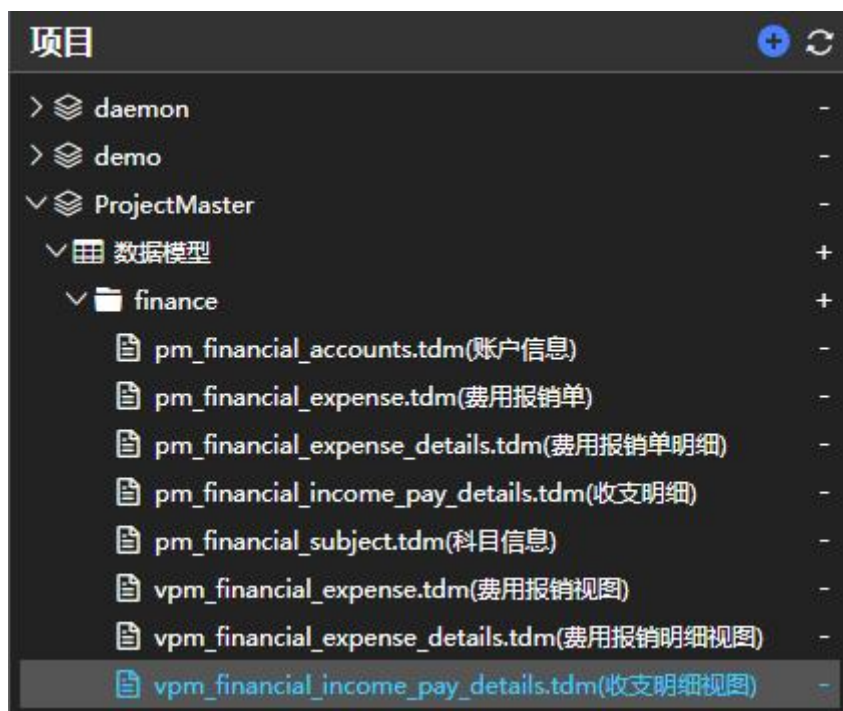
在此可以统一管理当前项目的数据模型，数据模型对应的是数据库中的表或视图。通过数据模型文件，一是可以在开发的过程中快速获取表或视图的相关信息，二是可以在数据模型文件中配置一些常用的校验规则和关联关系，以便在对这些表或视图进行操作时能自动进行数据校验和关联更新。

各个项目的数据模型文件的存储路径为：tasgine/app/项目代码/model。

点击项目 **数据模型** 节点右侧的加号图标，可以新建模型文件夹，可以新建表或视图，也可以从数据库导入已有的表或视图：



点击项目的 **数据模型** 节点，可以展开该节点查看里面已创建好的模型文件夹和模型文件列表，如下图所示：



点击文件夹右侧的加号，可以继续在该文件夹中创建子文件夹，或新建、导入数据模型。

数据模型文件的扩展名为.tdm（TaskMsg Data Model 的简称），这是 **TaskBuilder** 特有的一种文件格式，支持用 **TaskBuilder** 进行可视化编辑和设计。

点击数据模型文件可以打开 **数据模型设计器** 对该文件进行编辑设置，如下图所示：



后续文档会对 **数据模型设计器** 进行详细介绍，在此就不再深入解析。

## 2.2.2 后台服务

在此可以统一管理当前项目的所有后台服务程序，后台服务是指在服务器主机上由 **任擎** 软件加载和运行的代码，主要用来连接数据库软件，实现增删改查等操作，另外，也可以实现服务器端文件的读写，连接其他系统实现数据交换等。

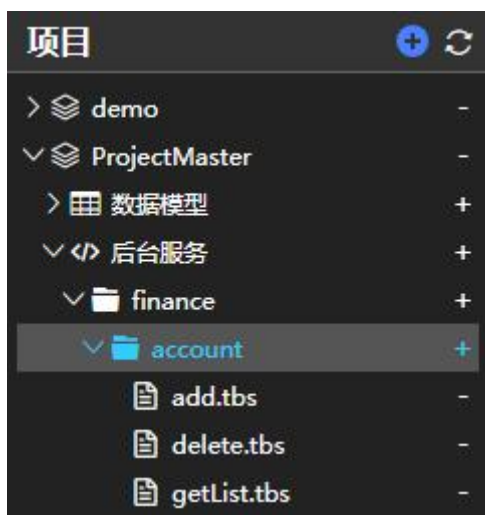
各个项目的后台服务文件的存储路径为：tasgine/app/项目代码/service。

点击项目 **后台服务** 节点右侧的加号，可以打开 **后台资源创建向导** 窗口，使用该向导可以创建后台文件夹和各种类型的后台服务：



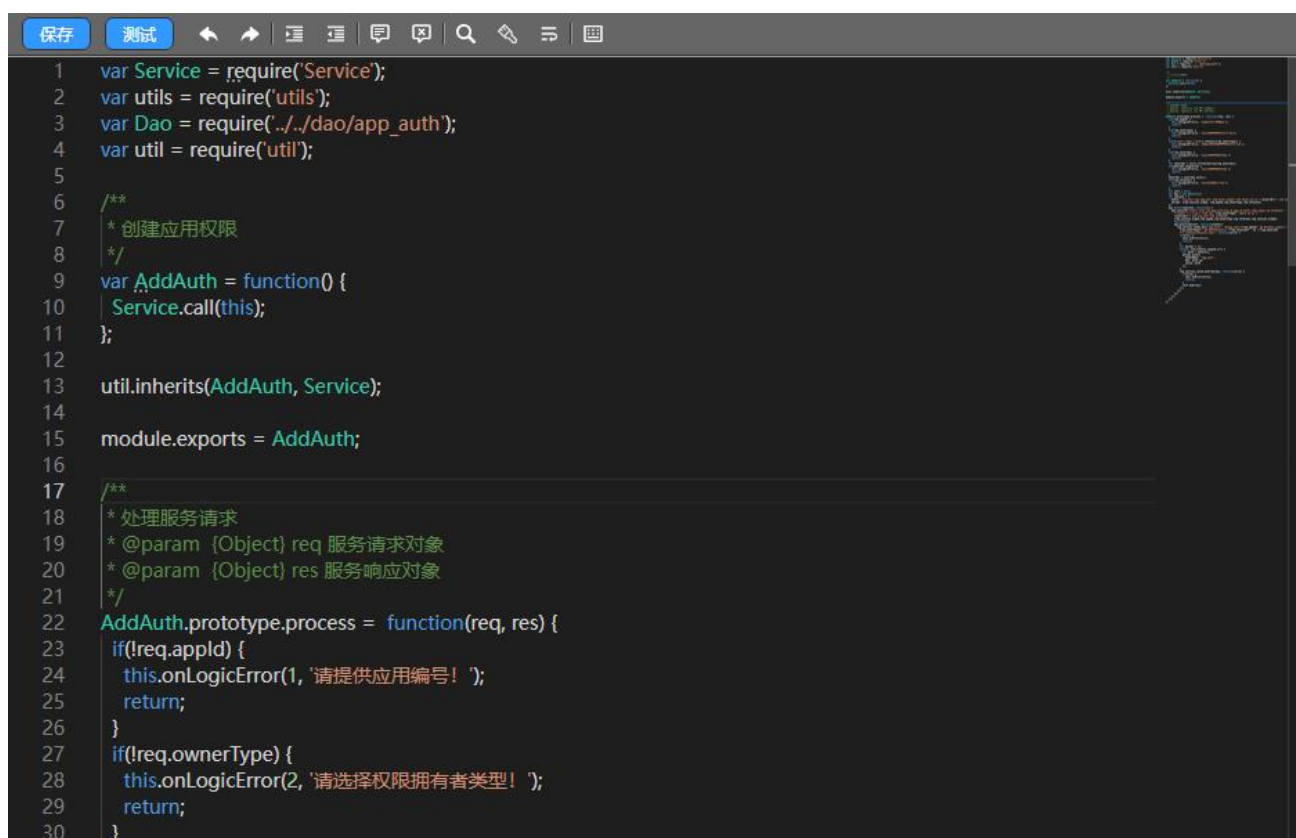
点击项目的 **后台服务** 节点，可以展开该节点查看里面已创建好的文件夹和文件列表，如下图所示：



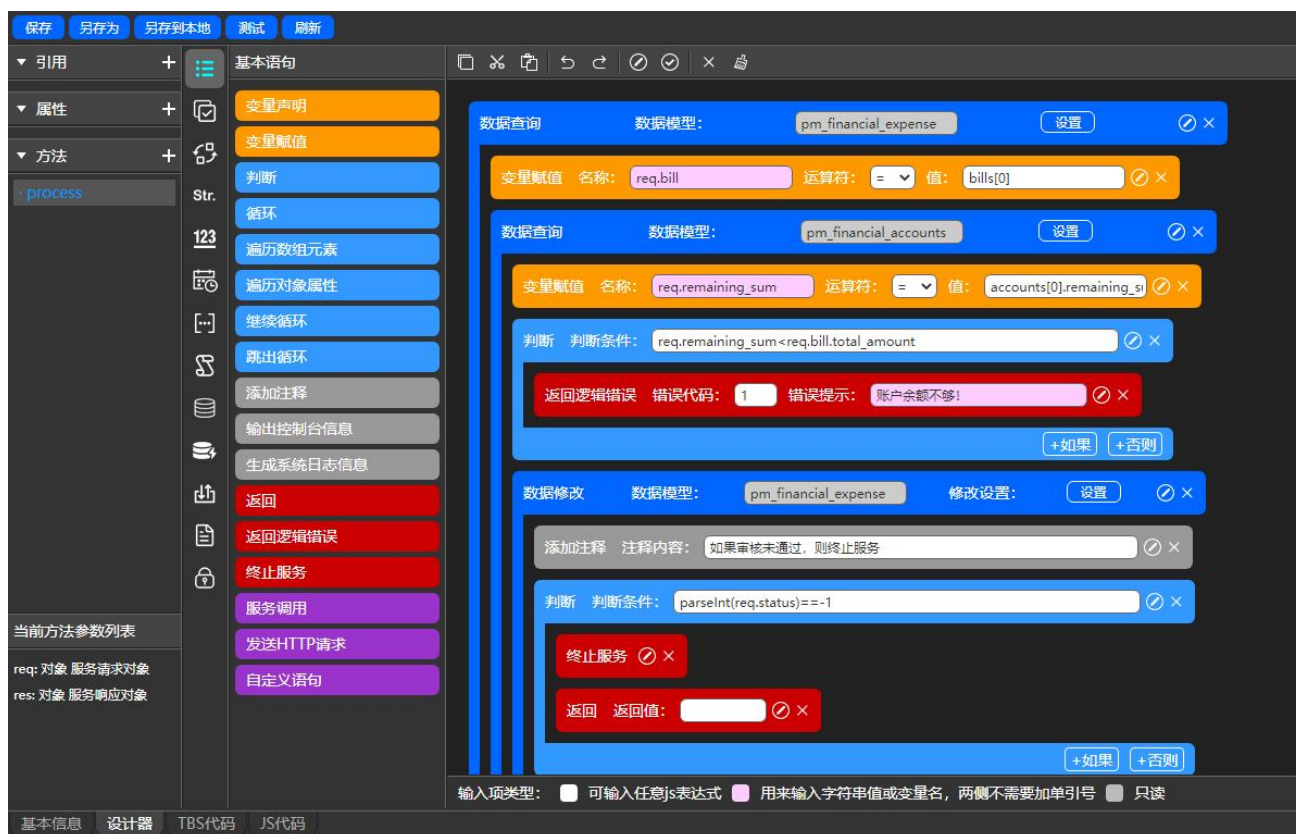


点击文件夹右侧的加号，可以继续在该文件夹中创建子文件夹，或新建后台服务。

后台服务文件支持 **js** 和 **tbs** 两种格式，js 格式是 JavaScript 源代码格式，如果在项目资源列表里点击 js 格式的后台服务文件，则会打开 **js 代码编辑器**，如下图所示：



**tbs** 格式（TaskMsg Background Service 的简称）是 **TaskBuilder** 特有的一种文件格式，支持用 **TaskBuilder** 进行可视化编辑和设计。点击 **tbs** 格式的后台服务文件，可以打开 **后台服务设计器** 对该文件进行编辑设置，如下图所示：

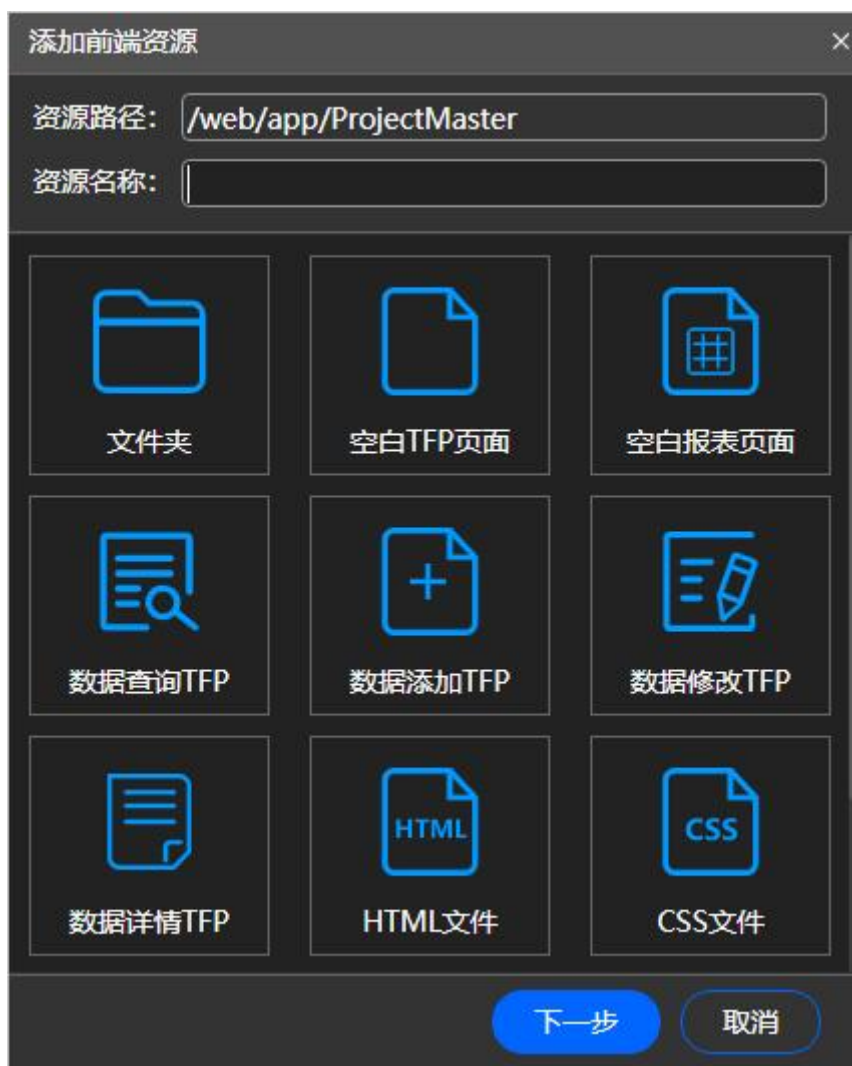


## 2.2.3 前端文件

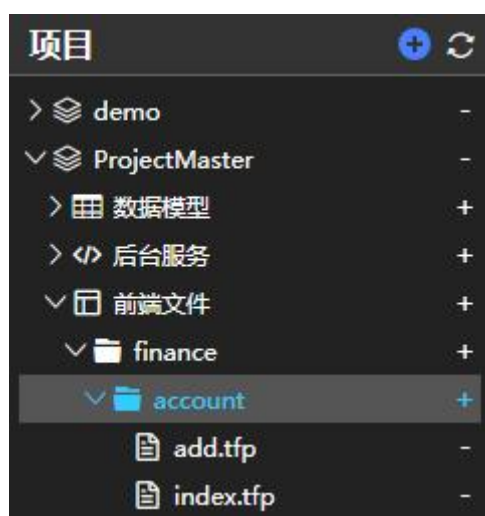
在此可以统一管理当前项目的各种前端文件，前端文件是指在用户浏览器中加载和运行的代码文件，主要用来录入和展示数据等，需要与后台服务配合，实现各种数据的提交和显示。

各个项目的前端文件的存储路径为：tasgine/web/app/项目代码。

点击项目 **前端文件** 节点右侧的加号，可以打开 **前端资源创建向导** 窗口，使用该向导可以创建前端文件夹和各种类型的前端文件。



点击项目的 **前端文件** 节点，可以展开该节点查看里面已创建好的文件夹和文件列表，如下图所示：

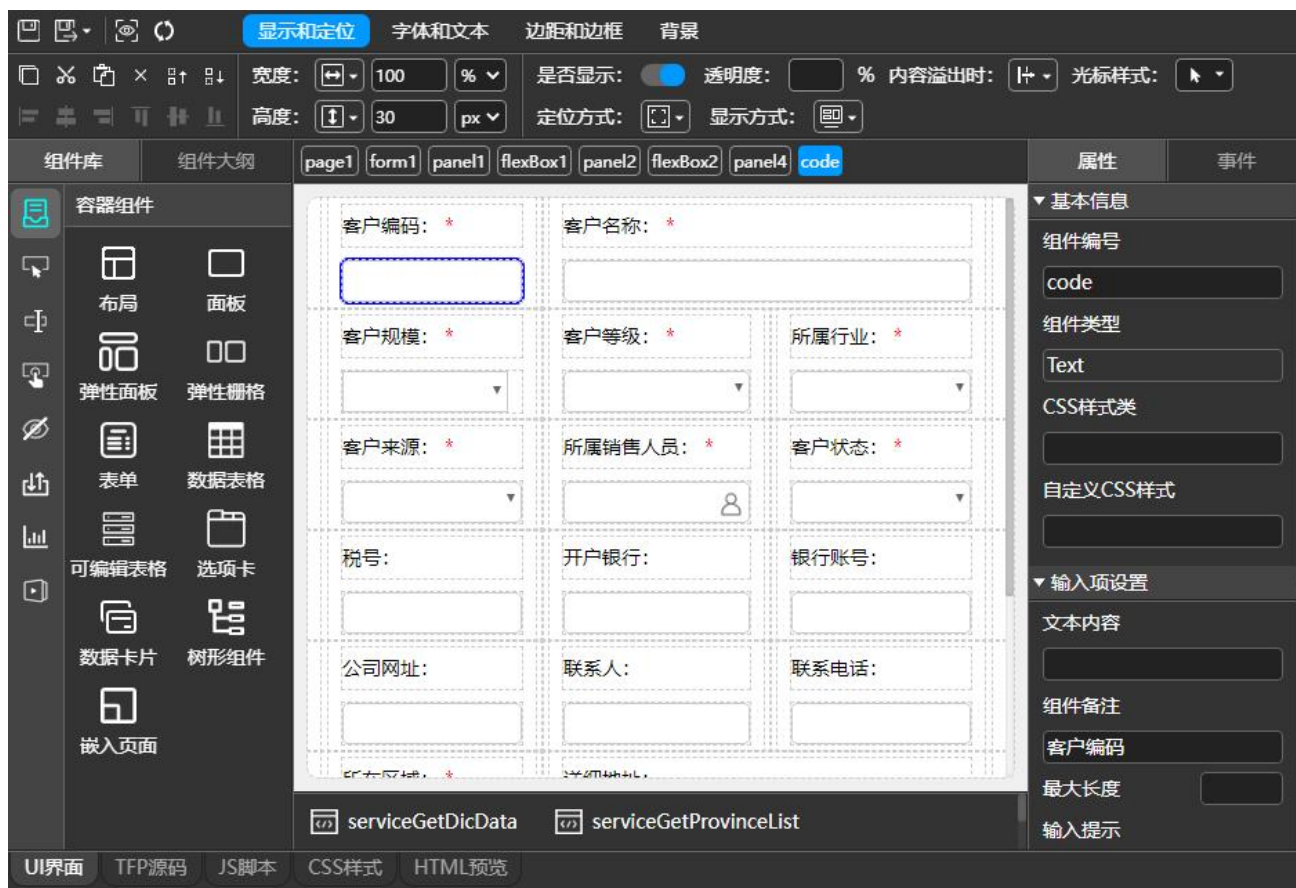


点击文件夹右侧的加号，可以继续在该文件夹中创建子文件夹，或新建前端文件。

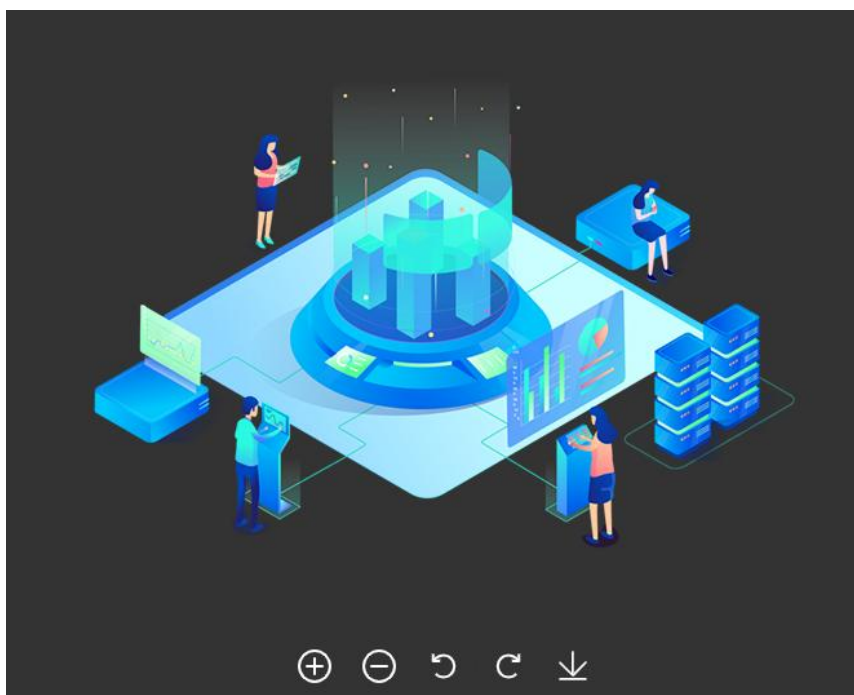
前端文件包括 html、css、js、tfp 等多种代码文件和 jpg、png 等图片文件。其中，**tfp**



格式（TaskMsg Front Page 的简称）是 **TaskBuilder** 特有的一种文件格式，支持用 **TaskBuilder** 进行可视化编辑和设计。点击 **tfp** 格式的前端文件，可以打开 **前端页面设计器** 对该文件进行编辑设置，如下图所示：



点击图片文件节点时，会打开 **图片浏览器**，如下图所示：



点击 html、css、js 等代码文件的节点时，会打开 **代码编辑器**，界面与上面后台服务里点击 js 格式的后台服务后打开的代码编辑器基本一致，只是少一个“测试”按钮，因为没有专门的前端代码测试工具。

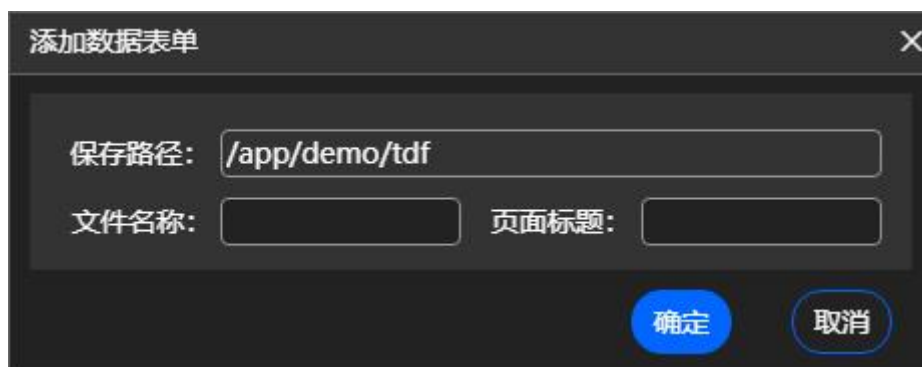
## 2.2.4 数据表单

在此可以统一管理当前项目的数据表单，数据表单文件的扩展名为 tdf（TaskMsg Data Form 的简称）。使用数据表单可以实现“零代码”开发，因为数据表单将数据模型、UI 界面和前后端业务逻辑全都结合到一起了，只要设计完表单界面，就会自动生成对应的数据模型，不需要事先创建，而且，在设计数据表单界面时，也可以配置各种校验规则，实现简单的业务逻辑校验。

相对前面介绍的“数据模型+后台服务+前端页面”各自分别配置的这种“低代码”开发方式，使用数据表单开发会更加简单便捷，但也正是因为其各层高度耦合，无法单独对某一层进行非常细致的设定，决定了其只适合开发一些相对简单的应用，不适合开发非常复杂的业务系统。

各个项目的数据表单文件的存储路径为：tasgine/app/项目代码/tdf。

点击项目 **数据表单** 节点右侧的加号，会弹出菜单，可以选择新建数据表单文件夹或新建数据表单，新建数据表单对话框界面如下：

该对话框标题为“添加数据表单”，右上角有关闭按钮。内部包含三个输入项：一个“保存路径”输入框，预填值为“/app/demo/tdf”；以及并列的“文件名称”和“页面标题”两个空输入框。底部右侧有两个按钮，分别为“确定”和“取消”。

输入文件名和页面标题后，点确定，即可创建一个新的数据表单，并会打开数据表单设计器显示该表单的设计界面，如下图所示：

在此可以设计表单的界面，具体如何使用后续会有详细介绍。

## 2.2.5 微信小程序

在此可以统一管理当前项目的微信小程序的相关资源文件，一个项目可以创建多个小程序，每个小程序里面又可以包含多个文件夹和 wxfp 文件，wxfp（WeiXin Front Page 的简称）文件是 TaskBuilder 特有的一种文件格式，用来设计微信小程序的界面，支持在 TaskBuilder 中可视化编辑，可以与 TaskBuilder 开发的后台服务配合，实现完整的、基于 C/S 模式的、复杂的微信小程序应用。

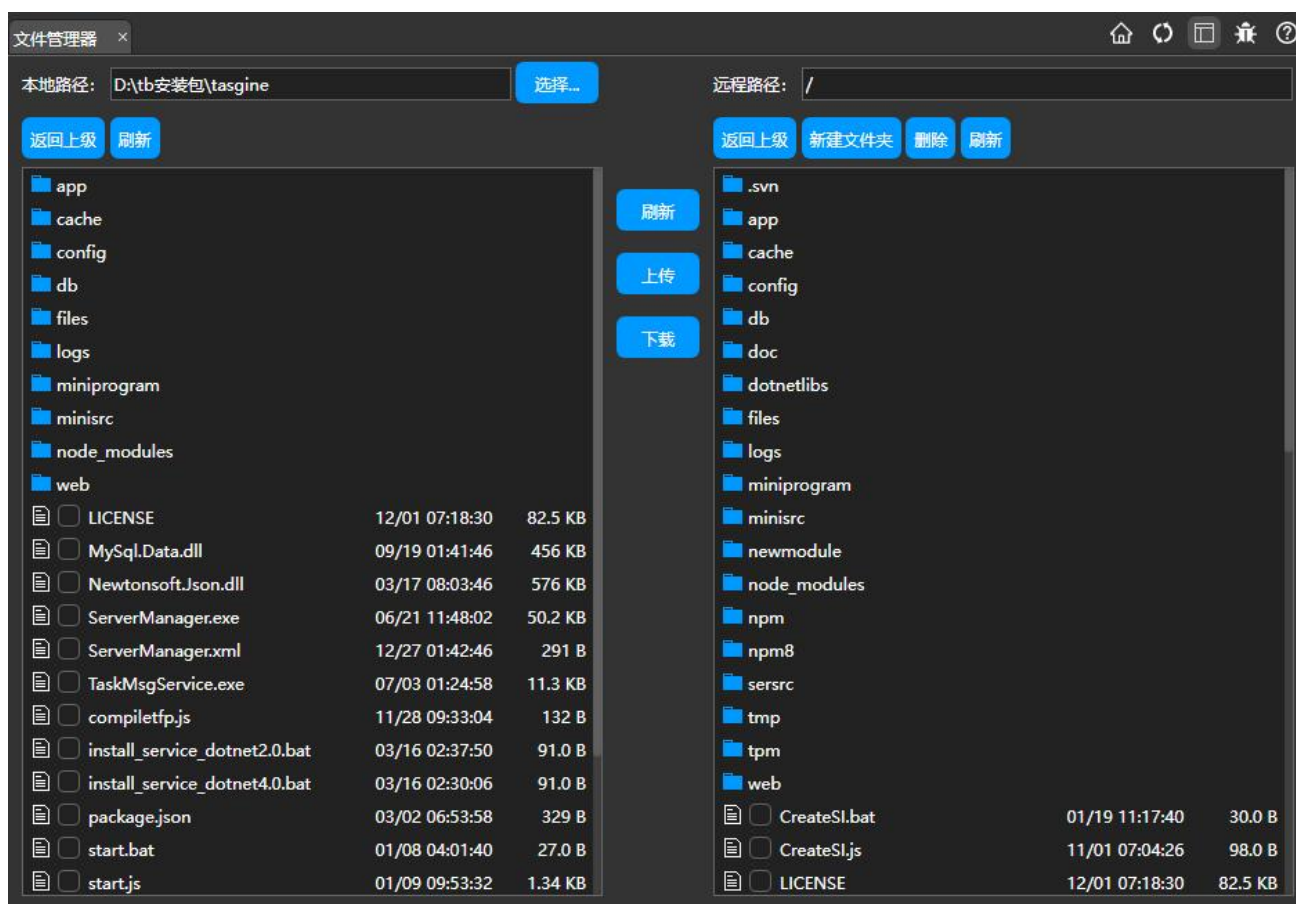
各个项目的微信小程序文件存储路径为：tasgine/minisrc/项目代码。

点击 **微信小程序** 节点右侧的加号，可以打开 **新建微信小程序** 窗口，在该窗口内输入小程序相关的信息，即可创建微信小程序应用，如下图所示：

在 TaskBuilder 里创建微信小程序之前，需要先去 **微信公众平台** 注册开通微信小程序账号，把该账号的 AppID 和 AppSecret 输入到上面的窗口内，才能完成新建；否则如果在这里随意输入 AppID 和 AppSecret，在后续打包发布小程序时，是无法通过 **微信公众平台** 验证的。如何进行微信小程序的开发，后续文档会有详细介绍，在此就不深入解析。

## 2.3 文件同步工具

在进行软件开发时，开发者一般会在自己电脑上搭建一个开发环境，先在自己电脑上进行开发和测试，没有问题后，再通过远程桌面、ftp 或其他工具把代码发布到正式服务器上。为了方便开发者同步本地开发环境和正式服务器上的文件，TaskBuilder 提供了一个文件同步工具，点击系统侧边栏上的文件管理图标 ，即可打开该工具，界面如下图所示：



在该工具的左侧，用来显示 TaskBuilder 所在电脑系统内指定文件夹的信息。刚打开时左侧列表是空的，需要点击上部的选择按钮选择一个本地电脑的文件夹，才能在左侧列表里看到这个文件夹里的子文件夹和文件信息。可以双击列表里的文件夹进入该文件夹，可以点击上部

的“返回上级”按钮退回到上级文件夹；可以点击“刷新”按钮重新加载当前文件夹的信息。

在该工具的右侧，用来显示 TaskBuilder 当前连接的任擎服务器上的文件夹和文件信息，默认显示任擎服务器根目录的内容。可以双击列表里的文件夹进入该文件夹，可以点击上部的“返回上级”按钮退回到上级文件夹；点击“新建文件夹”按钮可以新建一个文件夹；点击删除按钮，可以删除选中的文件；点击“刷新”按钮可以重新加载当前文件夹的信息。

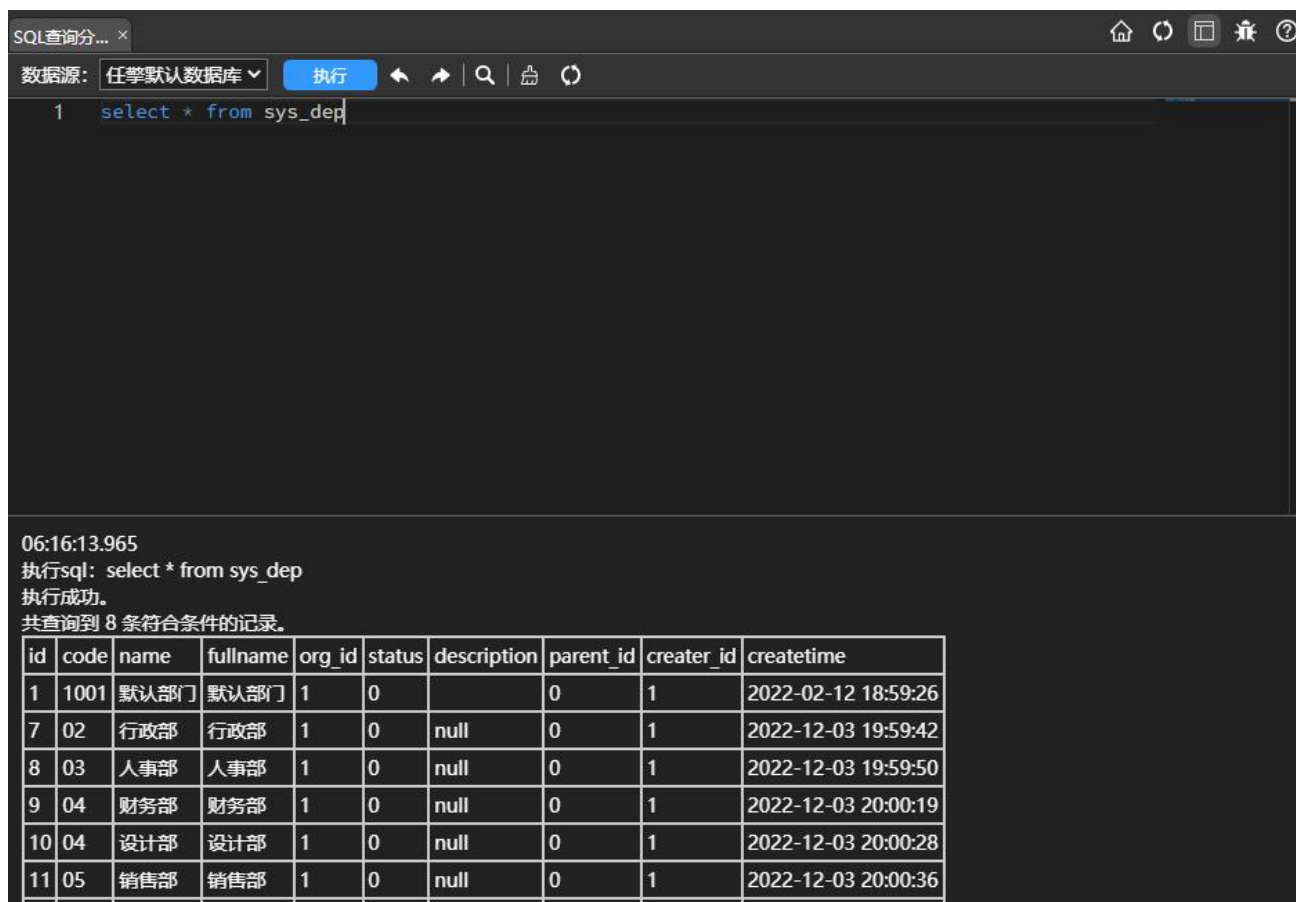
点击该工具中间的上传按钮，可以将左侧列表里选中的文件上传到服务器上，点击下载按钮，可以将右侧列表中选中的文件下载到本地电脑上。

**注意：**暂时不支持文件夹的上传下载。

## 2.4 SQL 执行工具

为了方便开发者连接当前任擎服务器上配置的各个数据源对应的数据库进行相关操作，

TaskBuilder 提供了一个 SQL 执行工具，点击系统侧边栏里的执行 SQL 图标 ，即可打开该工具，界面如下图所示：





该工具从上至下分为三个区域：

- **工具栏**：在此可以选择要执行数据库操作的数据源（数据库），可以点击“执行”按钮执行 SQL 编辑区内的 SQL 语句。另外，还提供了撤销、重做、查找、清空和刷新等按钮。
- **SQL 编辑区**：在此可以编辑要执行的 SQL 语句，支持关键字高亮显示。点击上部工具栏里的“执行”按钮时，如果在编辑区内没有选中任何语句，则会执行整个编辑区内的所有 SQL 语句，如果选中了某一条 SQL 语句，则仅执行选中的这条语句。
- **执行结果查看区**：在此可以查看 SQL 语句的执行结果。如果是查询语句，会以表格的形式显示查询结果。

**注意**：为了防止误操作造成数据丢失或混乱，在使用该工具执行修改和删除语句时，必须设置限定条件，不允许在此进行全表更新和删除。

另外，使用该工具执行查询操作时，最多只返回 1000 条数据，不会返回所有数据，以免造成引擎服务器卡顿，如要查询或导出大量数据，请使用专门的数据库管理工具。

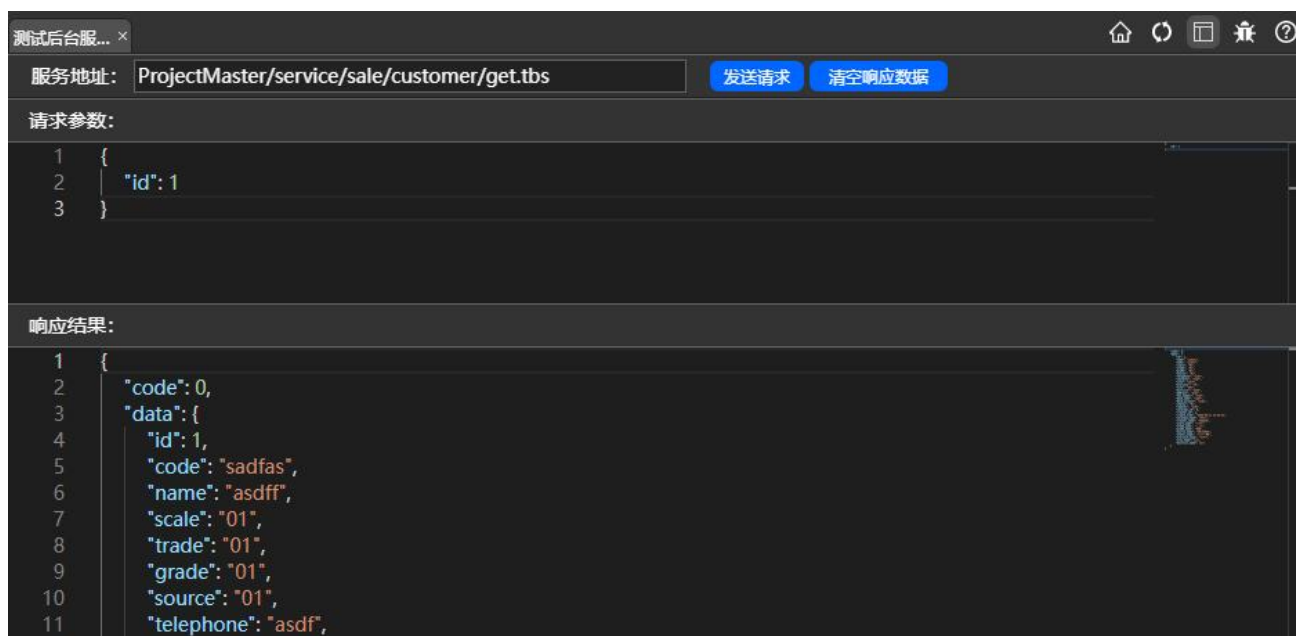
## 2.5 后台服务测试工具

为了方便开发者测试开发完的后台服务，TaskBuilder 提供了一个后台服务测试工具，点

击系统侧边栏里的请求服务图标



，即可打开该工具，界面如下图所示：



在该工具顶部，可以输入要测试的后台服务的地址，在中间区域的编辑器内设置好请求参

数（JSON 格式，如果没有，可以不用设置），然后点击“发送请求”按钮，即可模拟浏览器向引擎服务器发送服务请求，服务执行结果会显示在底部区域。

使用该工具，不需要通过前端页面就可以测试后台服务，方便开发测试或调试。使用该工具测试后台服务时，TaskBuilder 会采用服务器设置里当前服务器的用户和密码进行登录，如果该用户无权访问要测试的后台服务，会收到权限验证失败的错误信息。

## 2.6 登录后台管理系统

点击系统侧边栏上的后台管理系统图标 ，可以直接登录后台管理系统，在此可以查看配置的各种应用，也可以使用系统维护内的各项功能查看和设置系统的基础数据，包括组织结构、系统群组、应用、日志、附件、数据字典、编码规则和系统参数等。



## 2.7 进入开发者社区

为了方便 TaskBuilder 开发者互相之间交流，以及反馈问题，我们提供了一个开发者社区，点击 TaskBuilder 系统侧边栏上的社区按钮 ，即可进入开发者社区，可以在此查看各个板块内已经发布的信息，也可以发布新的内容，并查看回复。



### 3、服务器设置

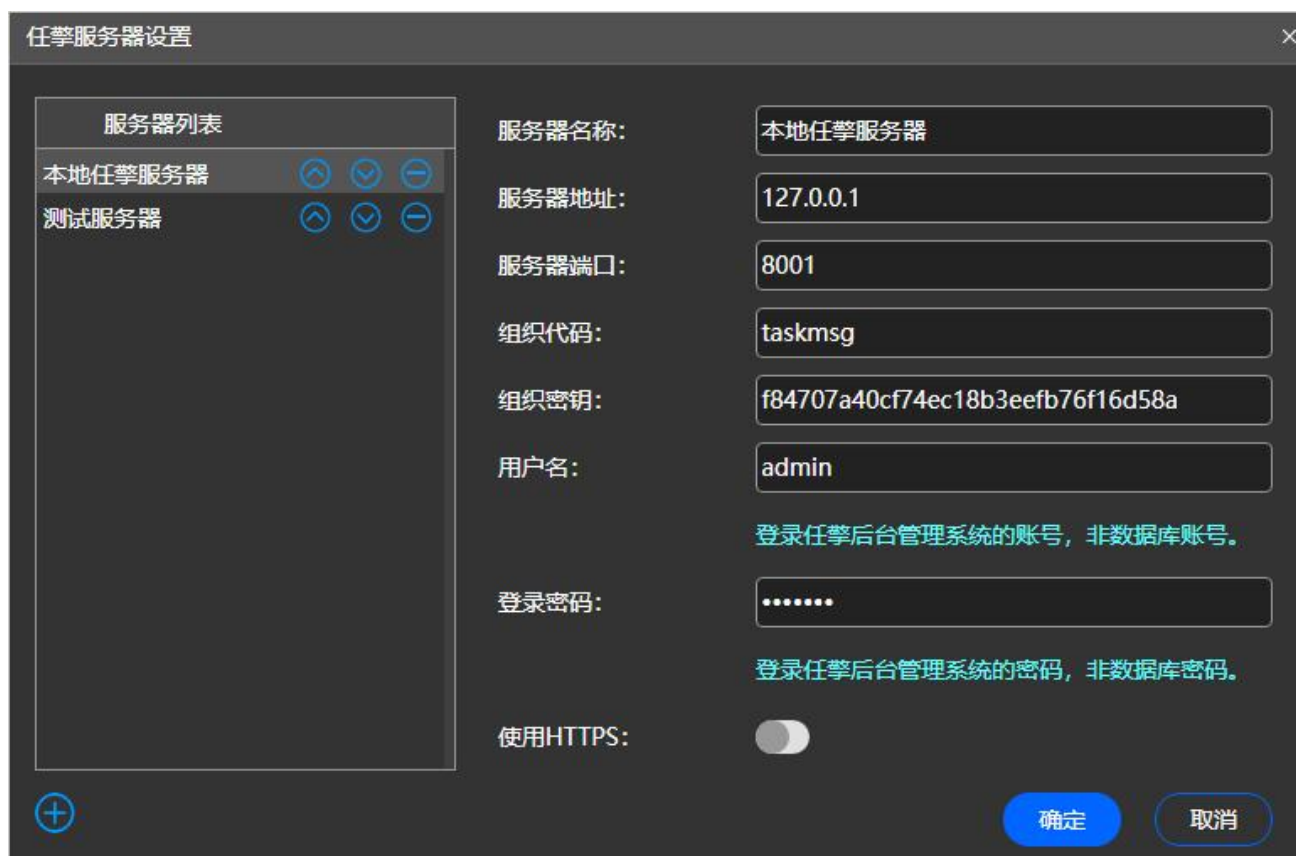
在使用 TaskBuilder 进行软件开发时，必须要先连接到任擎服务器（后续文档所说的服务器如果不特别说明，皆指任擎服务器）才能继续操作，因为使用 TaskBuilder 开发所需的数据模型、后台服务和前端页面等文件都存放在任擎服务器上，这些文件都需要通过任擎服务器才能解析和运行。所以，在使用 TaskBuilder 进行软件开发前，请先配置好任擎服务器。

对于很多开发者来说，一般会部署两套任擎服务器，一套安装在自己电脑上用来开发和测试，一套安装在公司或单位的服务器上，作为正式的运行环境；另外，有的开发者还部署了多套基于任擎的信息系统，为了方便访问，TaskBuilder 提供了专门的服务器设置功能，可以在 TaskBuilder 内配置多个任擎服务器的信息，从而可以实现用一个 TaskBuilder 连接多个任擎服务器进行开发，但 TaskBuilder 同时只能连一个任擎服务器，如果要访问另外一个服务器的资源，需要在服务器列表里选择这个服务器进行切换。

在第一次运行 TaskBuilder 时，会提供服务器配置向导，让开发者先配置好一个任擎服务器的信息，后续如果需要修改或添加其他服务器，可以点击 TaskBuilder 左上角的服务器列表右侧的设置按钮：



打开服务器设置界面进行设置，如下图所示：



在该窗口中，左侧显示已配置好的服务器列表，点击列表里各个服务器名称右侧的按钮，可以上移、下移或删除该服务器；点击窗口左下角的加号，可以添加新的服务器信息；点击服务器名称，可以在右侧查看和设置这个服务器的相关参数，包括以下参数：

- **服务器名称：**用来设置服务器的名称，以便在列表中查看和区分各个服务器，可以随意设置，但不能跟其他服务器重名，且只能是字母、汉字、数字和下划线。
- **地址：**用来设置任擎服务器的地址，可以是 IP 地址或域名，如果任擎服务器和 TaskBuilder 在同一台电脑上，这里的地址可以设置为：127.0.0.1。如果是域名，请不要添加 http:// 或 https:// 前缀，只需要填写域名本身即可，例如：server1.tasksmg.com。
- **端口：**用来设置任擎服务器的网络端口，默认为 8001，如果修改过任擎服务器的端口，则这里也需要进行相应的修改。如果任擎服务器和 TaskBuilder 不在同一台电脑上，请确定任擎服务器所在电脑是否启用了防火墙，防火墙是否开放了该网络端口，如果启用了防火墙、且没有开放任擎服务器的网络端口，则会导致 TaskBuilder 无法连接到该服务器。
- **组织代码：**用来设置任擎服务器的组织代码，默认为 taskmsg，如果注册过组织，则需要修改为注册的组织代码，可以在任擎后台管理系统的系统维护的系统信息里查看。
- **组织密钥：**用来设置任擎服务器的组织密钥，默认为

f84707a40cf74ec18b3eefb76f16d58a，如果注册过组织，则需要修改为注册的组织的密钥，可以在任擎后台管理系统的系统维护的系统信息里查看。请妥善保管该密钥，在使用任讯公司的相关服务时，都需要使用该密钥进行身份验证。

- **用户名：**用来设置登录任擎后台管理系统的用户名，不要设置为数据库的用户名。任擎服务器默认创建了一个 admin 账号，作为系统管理员的账号，在 TaskBuilder 里，可以直接使用该账号连接任擎，也可以使用其他账号。
- **登录密码：**用来设置通过上面的用户名登录任擎后台管理系统时的密码，admin 账号的默认密码为 taskmsg，如果使用任擎安装配置向导设置过密码，则在这里也要设置为这个密码，如果以后在任擎后台管理系统内修改了这个用户名的密码，则在这里也要同步修改。
- **是否使用 HTTPS 协议：**用来设置 TaskBuilder 连接任擎服务器时是否使用 HTTPS 协议，该设置需要与任擎服务器的对应设置相匹配。

打开 TaskBuilder 时，默认连接服务器列表里的第一个服务器，如果要连接其他服务器，可以点击服务器列表，选择其他服务器进行切换，切换后，TaskBuilder 左侧的系统资源列表会自动刷新，重新连接新选择的服务器并显示这个服务器上的资源信息。

## 4、系统辅助工具

在 TaskBuilder 主界面右上角提供了以下辅助工具的图标，点击后即可打开对应界面：



点击后会刷新整个 TaskBuilder 的界面，在刷新前，请先保存好界面中编辑过的相关数据，以免丢失。



点击后可以查看和反馈建议。



点击后可以查看在线帮助。



点击后可以查看更新公告。

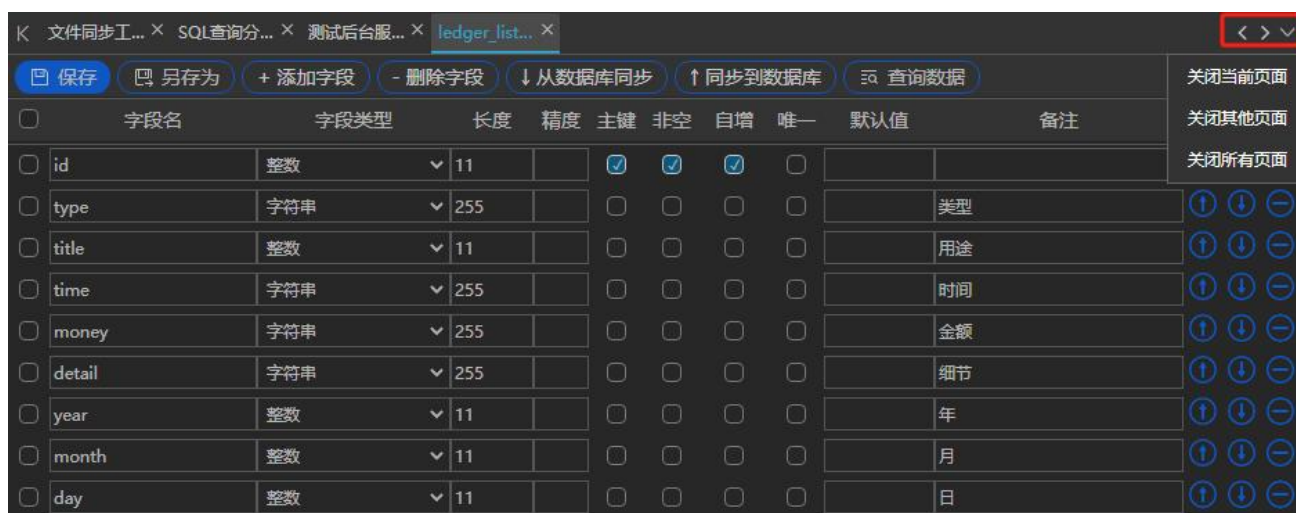


点击后可以切换是否显示或隐藏调试工具，该调试工具主要用来查看 TaskBuilder 本身在运行的过程中可能出现的一些错误信息，正常情况下不需要打开。

## 5、系统标签栏

使用 TaskBuilder 进行开发时，有一部分操作界面是以对话框的形式显示的，因为这些界面里要显示的内容相对比较少，例如创建各种资源文件的向导界面，而其他大部分功能的操作界面都是以标签页的形式显示在 TaskBuilder 主界面的操作区中，可以同时打开多个功能的操作页面，每个页面都会在系统标签栏显示一个对应的标签，标签内会显示该页面的标题。当打开了多个标签页时，只能显示其中某一个标签页的界面，点某个标签，可以将操作区界面切换到这个标签对应的页面。点击标签右上角的关闭图标，可以关闭该页面。

当系统标签栏内的标签很多时，可能会出现显示不下的情况，这时，可以点击系统标签栏右侧的“向左滚动”或“向右滚动”图标来显示更多标签。另外，点击系统标签栏最右侧的更多操作图标，会弹出一个下拉菜单，显示三个菜单项：关闭当前页面、关闭其他页面和关闭所有页面，点击后可以实现相应的效果。



## 6、系统状态栏

在 TaskBuilder 主界面的最底部是系统状态栏，左侧用来显示当前页面文件的路径，以便了解当前界面显示的是具体哪个文件的内容，中间显示最后一次操作的信息，右侧显示 TaskBuilder 版权信息和官网网址。



## 四、TaskBuilder 低代码开发基础知识介绍

在学习 TaskBuilder 的相关功能之前，我们先给大家介绍一些软件开发的基础知识，以便能更好地理解后续的内容，但都不会进行深入介绍，目的只是让没有接触过这些知识的读者能有一个初步的认识，如果要详细了解这些知识，请参阅其他专业书籍。如果大家对其中某些知识已经非常了解，也可以直接跳过。另外，本章的内容还会涉及到一些代码的示例，例如 HTML 代码、CSS 代码、JS 代码等，主要是为了讲解相关概念，在用 TaskBuilder 进行实际开发时，并不需要必须编写这些代码，大部分功能都能通过图形化的操作实现，所以，如果大家不想深入了解，也可以跳过这些代码示例，后面会介绍如何在 TaskBuilder 里用图形化的操作来实现这些功能。

### 1、数据库相关知识

#### 1.1 关系型数据库简介

TaskBuilder 主要用来开发基于关系型数据库（RDBMS）的增删改查类应用，关系型数据库是指使用关系模型进行数据组织和管理的数据库，其中数据以二维表格的形式表示，表格中的每一行表示一条数据记录，表格中的每一列表示一个数据属性。关系型数据库最早由 IBM 的研究员 E.F.Codd 在 20 世纪 70 年代提出，并成为了当今最流行的数据库类型之一，目前常用的关系型数据库有 SQL Server、MySQL、PostgreSQL、Oracle 等。除了关系型数据库外，还有键值数据库、文档数据库、时序数据库、图形数据库等，这些数据库本文档不会涉及，所以，如果没有特别声明，在后续内容中所说的数据库都是指关系型数据库。

可以将关系型数据库简单地理解为是由多个 Excel 表格构成的集合，只不过这些表格都是统一保存在一台服务器上，由专门的数据库软件进行管理，并且允许多个人通过网络同时访问，

但一般不会让用户直接操作数据库，那样一是不方便，二是不安全，因为没有完善的业务校验，很容易造成数据错乱，正常都是通过各种应用软件来对数据库的数据进行增删改查操作，可以进行操作权限控制和严格的业务校验，保障数据的准确性和完整性，而 TaskBuilder 就是专门开发这类软件的。

任擎可以同时连接多个关系型数据库，使用“安装配置向导”安装任擎服务器时，会创建一个系统数据库，里面包含了运行任擎所必需的表结构和初始数据，如果没有这个系统数据库，就无法使用 TaskBuilder 连接该任擎服务器进行低代码开发，也无法登录任擎后台管理系统进行管理和维护。如果要连接其他数据库，可以通过 TaskBuilder 里的数据源管理进行添加和设置。任擎服务器目前已支持 MySQL、PostgreSQL、Microsoft SQL Server 和人大金仓数据库，其中 MySQL 和 Microsoft SQL Server 提供了系统初始化 SQL 语句，其他数据库暂时未提供，但可以通过数据源连接访问。

## 1.2 为什么要用数据库？

在很多企事业单位中，现在大家主要用 Excel 记录和分享各种数据，Excel 确实简单易用，但它也存在一些问题：

- Excel 文档是一个单机文件，同一时间、同一个 Excel 文档只能在一台电脑上由一个用户访问和修改，无法同时多人查看和操作，如果同一个 Excel 文档需要多人查看和填报时，就非常不方便，需要挨个操作后，再由专人汇总，效率很低，还容易出错；
- Excel 文档的编辑权限无法按用户或角色进行精确控制，只要拿到 Excel 文档，只要没设置密码或只读，任何人都可以随意编辑里面的任何内容；
- 用 Excel 填报数据时，无法严格控制数据的格式和业务逻辑，导致用户填报上来的数据还需要人工整理，浪费人力；

- 有不少业务涉及到多种类型的数据，如果这些数据分别保存在多个 Excel 文件里，要想对这些数据进行关联查询和统计会很困难；
- 当数据量特别大时，例如上百万条记录，打开 Excel 文件会很慢，甚至无法操作。  
如果采用数据库，上面这些问题都会迎刃而解：
- 数据库系统安装在服务器上，多个用户可以通过应用系统同时连接数据库进行操作；
- 用户不能直接对数据库进行操作，必须经过应用系统提供的相关功能模块才能访问，而对这些功能模块的访问，在应用系统里是可以按用户或角色进行精确的权限控制的。
- 通过应用系统添加、修改或删除数据库中的数据时，可以在应用系统中进行严格的数据格式校验和逻辑判断，不满足要求的操作无法执行，可以有效保障数据的准确性和完整性；
- 如果所有业务数据都保存到数据库中，要关联查询和统计非常方便；
- 数据库软件是专门用来存储和管理数据的，主流的关系型数据库单表都可以支持存储上亿条记录，用户通过应用系统访问数据库时，可以使用 SQL 语句仅查询想要的数据库，数据量很大时，查询和增删改操作的速度要比 Excel 快很多。

从上面的对比可以看出，数据库在处理数据时相对于 Excel 有很大优势，尤其是需要多人填报和访问的情况，但原来由于开发这种操作数据库的软件需要专业的编程知识，对于普通办公人员来说太难了，所以大家没有办法，只能用 Excel 处理，而现在，随着低代码开发技术的出现，即使没有任何编程基础的普通办公人员，也可以通过简单的学习，就可以使用低代码开发工具快速开发出各种数据库增删改查的应用，从而大幅提升工作效率。

### 1.3 数据库中的表和视图

表和视图是数据库中最基本的两种对象，下面是对它们的简要介绍：



### 1.3.1 表 (table)

数据库中的表可以简单理解为类似 Excel 文件中的工作表，是一个二维表格，在表里面可以定义多个字段（相当于表格的列），每个字段可以设置具体的数据类型、长度、是否非空、是否唯一等限定条件，然后就可以使用应用软件对表进行添加、查询、修改和删除等操作，表里面可以存储任意多行数据，后续我们用 TaskBuilder 进行应用开发，主要就是围绕数据库中的表进行增删改查操作。

相对 Excel 的工作表，数据库中的表对数据的要求更加严格和规范，包括以下几点：

- 必须先定义好字段（列），无法在没有定义的字段里录入数据；
- 添加或修改数据时，必须满足所有字段的各种限定条件才能录入，例如数据类型、长度、非空、唯一等；
- 没有空行的概念，除非所有字段都可以为空，否则表中的每一行都是满足要求的数据，表里不存在空白的行；

可以说 Excel 是将数据存储和数据展示融为一体了，而且更加倾向于为数据展示服务，大量的功能都是为了方便用户查看和编辑数据，这种模式适合对数据格式要求不太严格的场合。而数据库是专门用来存储数据的，要确保数据的正确性和完整性，适合存储结构清晰的数据，它不关注数据如何展示，数据展示由专门的应用软件来实现，所以两者的定位不一样。

### 1.3.2 视图 (view)

按照关系型数据库的设计范式要求，一个数据库表中不能包含已在其它表中定义的非主关键字信息。例如，员工信息表中存储了部门编号后就不能再将部门名称、部门简介等与部门有关的其他信息再加入到员工信息表中，因为这些信息在部门信息表中已经存在了，如果在用户信息表里再保存一份，就会造成大量的数据冗余。当我们要查询包含部门信息的员工列表时，

可以编写 SQL（数据库查询语言，后面会专门介绍）语句，通过员工信息表里的部门编号关联部门信息表里的部门编号查询对应的部门信息，例如：

```
SELECT a.id AS user_id,a.name AS user_name,a.dep_id,b.name AS dep_name  
FROM sys_user a,sys_dep b WHERE a.dep_id=b.id;
```

这种多表关联查询在后续的开发中会经常用到，例如查询销售订单时需要关联查询客户信息，查询订单明细时，需要关联查询物料信息等。

如果某个关联查询需要在多个地方用到，则建议使用视图来实现，视图就是把上面的这种关联查询的 SQL 语句保存到数据库中，构成一个虚拟的表，后续在使用时，可以把这个视图当成一个表来用，但视图只能用于查询，不能对视图里的数据进行添加、修改和删除操作，因为视图不是实际存在的物理表。

将上面查询用户信息的语句创建为视图的 SQL 语句示例如下：

```
CREATE VIEW vsys_user AS  
  
SELECT a.id AS user_id,a.name AS user_name,a.dep_id,b.name AS dep_name  
FROM sys_user a,sys_dep b WHERE a.dep_id=b.id;
```

视图创建好后，后续查询这种多表关联的数据就很方便了，就像从一个表里查询数据一样，示例 SQL 语句如下：

```
SELECT user_id,user_name,dep_id,dep_name FROM vsys_user;
```

TaskBuilder 提供了创建数据库视图的工具，可以通过可视化的操作实现视图的创建和设置，后续会有章节进行详细介绍。

## 1.4 SQL 语言

### 1.4.1 SQL 语言简介

SQL 是 Structured Query Language（结构化查询语言）的缩写，是关系型数据库管理系统的标准语言。SQL 语句用于执行各种数据处理任务，如更新数据库中的数据，或从数据库中检索数据等。

使用 TaskBuilder 进行低代码开发时，大部分的数据库操作不需要编写 SQL 语句，直接通过可视化的界面进行配置即可，在后台服务运行时，引擎会自动根据这些操作的配置信息编译为相应的 SQL 语句然后提交给数据库执行，如果有特别复杂的数据库操作，TaskBuilder 也支持直接编写和执行 SQL 语句。

### 1.4.2 常见 SQL 语句类型

我们经常使用的 SQL 语句类型包括以下几种：

- **数据定义语言 (DDL)**: 用来实现库、表、视图、函数、存储过程等数据库对象的创建和删除等操作。
- **数据操作语言 (DML)**: 用来对数据库中表的数据记录进行增删改操作，包括 INSERT（添加数据）语句、UPDATE（修改数据）语句和 DELETE（删除数据）语句。
- **数据查询语言 (DQL)**: 用来查询数据库中表和视图的数据记录，其基本结构是使用 SELECT 子句、FROM 子句和 WHERE 子句的组合来查询一条或多条数据。

后续会对 SQL 的常见语句和语法进行详细介绍，这里就不赘述了。

## 2、JavaScript 语言

### 2.1 JavaScript 语言简介

JavaScript 是一种脚本语言，简称 js，在 1995 年由 Netscape 公司的 Brendan Eich，在网景导航者浏览器上首次设计实现而成。因为 Netscape 与 Sun 合作，Netscape 管理层希望它外观看起来像 Java，因此取名为 JavaScript。JavaScript 语言原来主要运行在 Web 浏览器环境中（前端），用来控制网页元素实现一些互动效果，目前市场上所有 Web 浏览器都支

持 JavaScript，后来随着 Node.js 的横空出世，让 JavaScript 在服务器端（后端或后台）也能运行了，这样就能做到只要掌握一种编程语言，就可以进行 Web 系统的前后端开发，不必再像以前那样，前端用 JavaScript，后端还得用 PHP、Java、C#等其他语言，大大降低了开发 Web 系统的难度，也正是看中这一点，TaskBuilder 低代码开发平台默认前后端都采用的 JavaScript 语言进行开发，但这里不展开做详细的介绍，请参阅专门介绍 JavaScript 的文章。

## 2.2 JavaScript 语言的特点

JavaScript 脚本语言具有以下特点：

- (1) **解释型语言。**JavaScript 是一种解释型的脚本语言，C、C++ 等语言需要先编译后执行，而 JavaScript 是在程序的运行过程中逐行进行解释，不需要事先编译。
- (2) **基于对象。**JavaScript 是一种基于对象的脚本语言，它不仅可以创建对象，也能使用现有的对象。
- (3) **简单。**JavaScript 语言采用的是弱类型的变量类型，对使用的数据类型未做出严格的要求，是基于 Java 基本语句和控制的脚本语言，其设计简单紧凑。
- (4) **跨平台性。**JavaScript 脚本语言不依赖于操作系统，仅需要浏览器的支持或 node.js 运行时环境。因此一个 JavaScript 脚本在编写后可以带到任意机器上使用。

## 2.3 JavaScript 语言的标准

发展初期，JavaScript 的标准并未确定，同期有 Netscape 的 JavaScript，微软的 JScript 和 CEnvi 的 ScriptEase 三足鼎立。为了互用性，1997 年，在 ECMA（欧洲计算机制造商协会）的协调下，由 Netscape、Sun、微软、Borland 组成的工作组制定了 JavaScript 语言统一的标准：ECMA-262。截至 2012 年，所有浏览器都完整的支持 ECMAScript 5.1（简称 ES5），

2015 年 6 月 17 日，ECMA 国际组织发布了 ECMAScript 的第六版，该版本正式名称为 ECMAScript 2015，但通常被称为 ECMAScript 6（简称 ES6）或者 ES2015。

相对于 ES5，ES6 标准定义的 JavaScript 功能有了大幅的增强，尤其是模块化编程，它可以将代码分割为独立的模块，每个模块都有自己的作用域，以及对外暴露的接口。这样的模块化方式可以提高代码的可维护性和可重用性，对于大型的、复杂的应用，建议采用 ES6 标准开发。TaskBuilder 里的前端 tfp 组件就是采用的 ES6 标准编写的，但为了兼容各种浏览器，在运行时，引擎会将组件代码编译成 ES5 标准的格式。

## 2.4 JavaScript 语言的模块化规范

在 JavaScript 语言刚刚诞生时，是为了实现一些简单的页面互动操作，但随着浏览器的不断发展，js 越来越被重视起来，可以用它实现较为复杂的功能。这个时候开发者为了维护方便，会把不同功能的模块抽离出来写入单独的 js 文件，但是当项目更为复杂的时候，html 可能会引入很多个 js 文件，而这个时候就会出现命名冲突，污染作用域等一系列问题，这个时候模块化的概念及实现方法应运而生。

JavaScript 的模块化规范除了 ES6 以外，还有 Node.js 支持的 CommonJS 规范，CommonJS 是最早提出的 JavaScript 模块化规范，主要用于 Node.js 环境，后来才有的 ES6 模块化规范，这两种规范的详细区别请参考相关文档。

目前，我们引擎采用的 Node.js 版本对这两种规范都支持，但由于我们 2014 年在开发引擎时，ES6 标准还未制定，所以，引擎采用的是 CommonJS 规范，包括引擎的核心代码以及开发者用 TaskBuilder 开发的后台服务，都是按 CommonJS 规范编译成相应的 js 代码。

## 2.5 JavaScript 语言的运行环境

在用 TaskBuilder 开发的应用中，不同文件中的 JavaScript 代码会运行在不同的运行环境中，在这些运行时环境中，js 的基本语法都是一样的，但每种环境各自都有不同的适用场

景和一些特殊功能，具体见下表：

| 运行时环境            | 适用场景                                                                           | 常见用途                                                           |
|------------------|--------------------------------------------------------------------------------|----------------------------------------------------------------|
| 引擎服务器            | 用来运行后台服务的 JavaScript 代码。                                                       | 可以进行数据库增删改查、读写服务器端文件等操作。                                       |
| Web 浏览器          | 用来运行前端页面中的 JavaScript 代码。电脑端页面由电脑端 Web 浏览器运行，手机端 H5 页面由手机端浏览器或手机 APP 内置的浏览器运行。 | 可以访问和控制页面中的组件或 HTML 元素。出于安全考虑，浏览器不允许网页内的 js 访问电脑或手机上的文件、硬件等资源。 |
| 任讯 APP、阿里钉钉、企业微信 | 用来运行专门为这几款 APP 开发的手机端页面中的 JavaScript 代码，本质上也是通过嵌入的浏览器来运行，只不过各自都提供了一些特殊的扩展功能。   | 除了可以像浏览器中的 js 一样访问和控制页面元素外，还提供了拍照、扫码、定位等扩展功能。                  |
| 微信               | 用来运行微信小程序中的 JavaScript 代码，页面元素与 H5 页面不完全一样，有自己的一套组件和访问控制机制。                    | 可以访问和控制小程序页面中的组件，并提供了读取微信用户信息、拍照、扫码等扩展功能。                      |

## 2.6 JSON 数据格式

### 2.6.1 JSON 简介

JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 ECMAScript 的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。JSON 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升了网络传输效率。简单地说，JSON 可以将 JavaScript 对象中表示的一组数据转换为字符串，然后就可以在网络或者程序之间轻松地传递这个字符串，并在需要的时候将它还原为各编程语言所支持的数据格式。

由于 JavaScript 语言默认支持 JSON，所以用 TaskBuilder 开发的应用前后端传递数据的格式就是采用的 JSON 格式。

## 2.6.2 JSON 数据格式的构成

JSON 是一个序列化的对象或数组。

JSON 对象由花括号 { } 括起来的多个键值对构成，数据结构为 {key1: value1, key2: value2, ...}，键只能是字符串，值可以是对象、数组、数字、字符串或者三个字面值(false、null、true)中的一个，键和值用英文冒号 (:) 分隔，多个键值对用英文逗号分割，例如：

```
{  
  "name": "张三",  
  "age": 18,  
  "disabled": false  
}
```

JSON 数组是由方括号 [ ] 括起来的一组值构成，数据结构为 ["java", "javascript", "vb", ...]，多个值用英文逗号分隔，例如：

```
[3, 1, 4, 1, 5, 9, 2, 6]
```

JSON 对象里的某个键值对的值可以是对象或数组，例如：

```
{  
  "name": "张三",  
  "age": 18,  
  "disabled": false,  
  "address": {  
    "city": "北京",  
    "zip-code": "10000"  
  },  
  "roles": [3, 4, 7]  
}
```

数组的值也可以是 JSON 对象，例如：

```
[{"name": "张三", "sex": "男"}, {"name": "李四", "sex": "女"}]
```



## 2.6.3 JSON 与 JS 对象的关系

JSON 是 JS 对象的字符串表示法，它使用文本表示一个 JS 对象的信息，本质是一个字符串，例如以下代码：

```
var obj = {a: 'Hello', b: 'World'}; //这是一个 JS 对象
```

```
var json = '{"a": "Hello", "b": "World"}'; //这是一个 JSON 字符串
```

要实现从 JSON 字符串转换为 JS 对象，使用 JSON.parse() 方法：

```
var obj = JSON.parse('{"a": "Hello", "b": "World"}'); //结果是 {a: 'Hello', b: 'World'}
```

要实现从 JS 对象转换为 JSON 字符串，使用 JSON.stringify() 方法：

```
var json = JSON.stringify({a: 'Hello', b: 'World'}); //结果是 '{"a": "Hello", "b":  
"World"}'
```

## 3、HTML 语言

### 3.1 HTML 简介

HTML 的全称为超文本标记语言，它包括一系列标签，通过这些标签可以将网络上的文档格式统一，使分散的 Internet 资源连接为一个逻辑整体。使用 HTML，将所需要表达的信息按某种规则写成 HTML 文件，通过 Web 浏览器来识别，并将这些 HTML 文件“翻译”成可以识别的信息，即我们所见到的网页。网页文件本身是一种文本文件，通过在文本文件中添加标记符，可以告诉浏览器如何显示其中的内容（如：文字如何处理，画面如何安排，图片如何显示等）。浏览器按顺序阅读网页文件，然后根据标记符解释和显示其标记的内容。

自 1990 年以来，HTML 就一直被用作万维网的信息表示语言，使用 HTML 描述的文件需要通过 web 浏览器才能显示出效果。HTML 是一种建立网页文件的语言，通过标记式的指

令(Tag)，将文字、图形、动画、声音、表格、链接、影像等内容显示出来。事实上，每一个 HTML 文档都是一种静态的网页文件，这个文件里面包含了 HTML 指令代码，这些指令代码并不是一种程序语言，只是一种排版网页中资料显示位置的标记结构语言，易学易懂，非常简单。HTML 的普遍应用就是带来了超文本的技术：通过单击鼠标从一个主题跳转到另一个主题，从一个页面跳转到另一个页面，实现与世界各地 Web 主机的文件链接。超文本传输协议规定了浏览器在运行 HTML 文档时所遵循的规则和进行的操作。

使用 TaskBuilder 开发的 tfp 前端页面，保存的格式为 JSON 格式，在有用户访问该页面时，会自动被引擎编译为 HTML 格式的文档，不需要开发人员手动编辑 HTML 内容。

## 3.2 HTML 的版本

HTML 历史上有如下版本：

- ① HTML 1.0：在 1993 年 6 月作为互联网工程工作小组(IETF)工作草案发布。
- ② HTML 2.0：1995 年 11 月作为 RFC 1866 发布，于 2000 年 6 月发布之后被宣布已经过时。
- ③ HTML 3.2：1997 年 1 月 14 日，W3C 推荐标准。
- ④ HTML 4.0：1997 年 12 月 18 日，W3C 推荐标准。
- ⑤ HTML 4.01（微小改进）：1999 年 12 月 24 日，W3C 推荐标准。
- ⑥ HTML 5：HTML5 是公认的下一代 Web 语言（简称 H5），极大地提升了 Web 在富媒体、富内容和富应用等方面的能力，被喻为终将改变移动互联网的重要推手。虽然现在电脑端和手机端的浏览器都支持 HTML5，但大家平时所说的 H5 页面一般都是指针对移动端开发的 HTML 页面。

目前几乎所有浏览器的最新版本都支持 HTML 5，包括 Chrome、Firefox、Safari、Edge 等，不支持 HTML 5 的浏览器不建议使用，例如 Internet Explorer 8 及以前的版本。

用 TaskBuilder 开发的前端页面采用的是 HTML5 标准。

### 3.3 HTML 文档的结构

HTML 文件以.htm 或.html 为扩展名，可以使用任何能够生成文本文件的编辑器来生成 HTML 文件，只用修改文件后缀即可。标准的超文本标记语言文件都具有一个基本的整体结构，标记一般都是成对出现（部分标记除外例如：`<br/>` `<img/>`）。

#### 3.3.1 文档标记

标记符`<html>`说明该文件是用超文本标记语言来描述的，它是文件的开头，而`</html>`标记则表示该文件的结尾，它们是超文本标记语言文件的开始标记和结尾标记。

#### 3.3.2 头部内容

`<head>` `</head>`这 2 个标记符分别表示 HTML 文档头部信息的开始和结尾。头部中包含的标记是页面的标题、序言、说明等内容，它本身不作为内容来显示，但影响网页显示的效果。头部中最常用的标记符是标题标记符`<title>`和`<meta>`标记符，其中标题标记符用于定义网页的标题，它的内容显示在网页窗口的标题栏中，网页标题可被浏览器用做书签和收藏清单。

#### 3.3.3 主体内容

`<body>` `</body>`这 2 个标记符分别表示 HTML 文档主体内容的开始和结尾，网页中显示的实际内容均包含在这 2 个标记符之间。

#### 3.3.4 文档元素

在文档主体标记中，可以定义需要显示的各种元素标记，常见的有：

- **标题:** `<h1></h1>`, 一共有 6 级, h 后面的数字代表级别, 数字越大, 标题级别越小;
- **文本段落:** `<p></p>`, 用来显示文本段落;
- **区块:** `<div></div>`, 用来定义一个区块, 里面可以再放置其他元素;
- **表单:** `<form></form>`, 用来定义一个数据表单, 里面可以放各种输入项;
- **输入项:** `<input>`, 用来定义输入项, 可以设置输入项的类型, 例如文本、密码等;
- **图片:** `<img>`, 用来定义一个图片元素;

更多详细的 HTML 标签说明请参阅相关文档。

### 3.3.5 HTML 文档示例

下面是一个最简单的 HTML 文档的内容:

```
<html>
  <head>
    <title>TaskBuilder 低代码开发入门</title>
  </head>
  <body>
    <p>Hello, TaskBuilder! </p>
  </body>
</html>
```

大家可以把上面的代码保存到一个文本文件中, 并将扩展名改为.html, 然后双击打开, 看看是不是能在浏览器里看到这个页面的内容。

## 4、CSS 样式

### 4.1 CSS 简介

CSS 层叠样式表(英文全称: Cascading Style Sheets)是一种用来控制 HTML 元素外观样式的计算机语言, CSS 不仅可以静态地修饰网页, 还可以配合脚本语言动态地对网页元素进行

样式格式化。

从 HTML 被发明开始，样式就以各种形式存在。不同的浏览器结合它们各自的样式语言为用户提供页面效果的控制。最初的 HTML 只包含很少的显示属性，随着 HTML 的发展，为了满足页面设计者的要求，HTML 添加了很多显示功能。但是随着这些功能的增加，HTML 变的越来越杂乱，而且 HTML 页面也越来越臃肿，于是 CSS 便诞生了。

CSS 为 HTML 标记语言提供了一种样式描述，定义了其中元素的显示方式。CSS 在 Web 设计领域是一个突破，利用它可以实现修改一个小的样式便能更新与之相关的所有页面元素。

## 4.2 CSS 的特点

相对于 HTML 提供的元素样式控制功能，CSS 具有以下特点：

### (1) 丰富的样式定义

CSS 提供了丰富的文档样式外观，以及设置文本和背景属性的能力；允许为任何元素创建边框，以及元素边框与其他元素间的距离，以及元素边框与元素内容间的距离；允许随意改变文本的大小写方式、修饰方式以及其他页面效果。

### (2) 易于使用和修改

CSS 可以将样式定义在 HTML 元素的 style 属性中，也可以将其定义在 HTML 文档的 header 部分，也可以将样式声明在一个专门的 CSS 文件中，以供 HTML 页面引用。总之，CSS 样式表可以将所有的样式声明统一存放，进行统一管理。

另外，可以将相同样式的元素进行归类，使用同一个样式进行定义，也可以将某个样式应用到所有同名的 HTML 标签中，也可以将一个 CSS 样式指定到某个页面元素中。如果要修改样式，我们只需要在样式列表中找到相应的样式声明进行修改。

### (3) 多页面应用

CSS 样式表可以单独存放在一个 CSS 文件中，这样我们就可以在多个页面中使用同一个 CSS 样式表。CSS 样式表理论上不属于任何页面文件，在任何页面文件中都可以将其引用。这样就可以实现多个页面风格的统一。

#### (4) 层叠

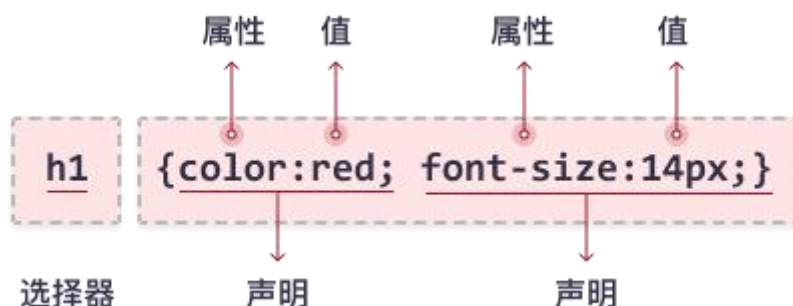
简单的说，层叠就是对一个元素多次设置同一个样式，这将使用最后一次设置的属性值。例如对一个站点中的多个页面使用了同一套 CSS 样式表，而某些页面中的某些元素想使用其他样式，就可以针对这些样式单独定义一个样式表应用到页面中。这些后来定义的样式将对前面的样式设置进行重写，在浏览器中看到的将是最后面设置的样式效果。

#### (5) 页面压缩

在使用 HTML 定义页面效果的网站中，往往需要大量或重复的表格和 font 元素形成各种规格的文字样式，这样做的后果就是会产生大量的 HTML 标签，从而使页面文件的大小增加。而将样式的声明单独放到 CSS 样式表中，可以大大的减小页面的体积，这样在加载页面时使用的也会大大的减少。另外，CSS 样式表的复用更大程序的缩减了页面的体积，减少下载的时间。

### 4.3 CSS 语法

CSS 规则集 (rule-set) 由选择器和声明块组成，如下图所示：



选择器指向您需要设置样式的 HTML 元素，声明块包含一条或多条用分号分隔的声明，

每条声明都包含一个 CSS 属性名称和一个值，以冒号分隔。多条 CSS 声明用分号分隔，声明块用花括号括起来。

下面的示例中，所有 `<p>` 元素都将居中对齐，并带有红色文本颜色：

```
p {  
  color: red;  
  text-align: center;  
}
```

例子解释：

`p` 是 CSS 中的选择器（它指向要设置样式的 HTML 元素：`<p>`）。

`color` 是属性，`red` 是属性值

`text-align` 是属性，`center` 是属性值

## 4.4 CSS 选择器

CSS 选择器用于“查找”（或选取）要设置样式的 HTML 元素，CSS 选择器有很多种类型，我们这里仅介绍几个最基本的选择器，更详细的内容请参考相关文档。

### 4.4.1 元素选择器

元素选择器根据元素名称来选择 HTML 元素，示例代码如下：

```
p {  
  text-align: center;  
  color: red;  
}
```

使用上面的样式选择器，页面上的所有 `<p>` 元素都将居中对齐，并带有红色文本颜色。

### 4.4.2 id 选择器

`id` 选择器使用 HTML 元素的 `id` 属性来选择特定元素，元素的 `id` 在页面中是唯一的，



因此 id 选择器用于选择一个唯一的元素。要选择具有特定 id 的元素，需要在选择器前面加一个井号（#），然后再写元素的 id，示例代码如下：

```
#para1 {  
    text-align: center;  
    color: red;  
}
```

以上 CSS 规则将应用于 id="para1" 的 HTML 元素。

### 4.4.3 类选择器

类选择器选择有特定 class 属性的 HTML 元素，在选择拥有特定 class 的元素时，需要在选择器前面写一个句点（.）字符，后面跟类名，示例代码如下：

```
.center {  
    text-align: center;  
    color: red;  
}
```

使用上面的样式规则，页面内所有带 class="center" 的 HTML 元素的文字将变为红色且居中对齐。

还可以指定只有特定的 HTML 元素会受类选择器的影响，示例代码如下：

```
p.center {  
    text-align: center;  
    color: red;  
}
```

使用上面的样式规则，只有具有 class="center" 的 <p> 元素的内容会居中对齐，且文字会变成红色。

HTML 元素也可以引用多个 CSS 样式类，示例代码如下：

```
<p class="center large">这个段落引用两个类。</p>
```

在这个例子中，<p> 元素将根据 class="center" 和 class="large" 这两个 CSS 样式类

进行样式设置。

#### 4.4.4 分组选择器

如果多个 CSS 选择器的样式定义相同，则可以使用分组选择器，示例代码如下：

```
p, #para1, .center {  
    text-align: center;  
    color: red;  
}
```

在上面这个例子中，p、 #para1、.center 这三个选择器具有相同的样式定义，所以可以共用一个分组选择器，不需要分别定义，这样可以节省代码。

### 4.5 CSS 的使用

CSS 样式有三种使用方法：

#### 4.5.1 外部 CSS

外部 CSS 样式是指在单独的 .css 文件中定义的 CSS 样式，可以在 HTML 页面的 head 部分用 <link> 元素引用外部样式文件，示例代码如下：

```
<html>  
  <head>  
    <link rel="stylesheet" type="text/css" href="mystyle.css">  
  </head>  
  <body>  
    <h1>这是一个标题</h1>  
    <p>这是一个段落。</p>  
  </body>  
</html>
```

使用外部 CSS 样式的好处：

- (1) 一个 CSS 样式文件可以给多个 HTML 文件使用，不用重复定义；

(2) 将 CSS 代码和 HTML 代码分离，易于维护；

## 4.5.2 内部 CSS

内部样式是指在 HTML 文档内部 head 部分的 <style> 元素中定义的 CSS 样式，示例代码如下：

```
<html>
  <head>
    <style>
      body {
        background-color: #F0F0F0;
      }
      h1 {
        color: #333333;
        margin-left: 20px;
      }
    </style>
  </head>
  <body>
    <h1>这是一个标题</h1>
    <p>这是一个段落。</p>
  </body>
</html>
```

## 4.5.3 行内 CSS

行内样式（也称内联样式）用于为单个 HTML 元素设置样式，如需使用行内样式，请将 style 属性添加到相关元素，style 属性可包含任何 CSS 样式，示例代码如下：

```
<html>
  <body>
    <h1 style="color:blue;text-align:center;">这是一个标题</h1>
    <p style="color:red;">这是一个段落。</p>
```

```
</body>  
</html>
```

后续章节会详细介绍如何在 TaskBuilder 中使用上面介绍的三种方式设置 CSS 样式。

## 5、Node.js

### 5.1 Node.js 简介

Node.js 发布于 2009 年 5 月，由 Ryan Dahl 开发，是一个基于 Chrome V8 引擎的 JavaScript 运行环境，使用了一个事件驱动、非阻塞式 I/O 模型，可以让 JavaScript 运行在服务端环境中，它的出现，让 JavaScript 成为与 Java、.Net、PHP 等服务端语言平起平坐的脚本语言。



Node.js 对 Chrome V8 引擎的一些特殊用例进行了优化，使得 V8 在非浏览器环境下运行得更好，V8 引擎执行 Javascript 的速度非常快，性能非常好，可以用于搭建响应速度快、易于扩展的网络应用。

V8 引擎本身使用了一些最新的编译技术，这使得用 Javascript 这类脚本语言编写出来的代码运行速度获得了极大提升，又节省了开发成本。对性能的苛求是 Node.js 的一个关键因素，Node.js 采用了一个称为“事件循环 (event loop)”的架构，使得编写可扩展性高的服务器变得既容易又安全。提高服务器性能的技巧有多种多样，Node.js 选择了一种既能提高性能，又能减低开发复杂度的架构，这是一个非常重要的特性，并发编程通常很复杂且布满地雷。Node.js 绕过了这些，但仍提供很好的性能。

引擎服务器就是基于 Node.js 开发，并增加了很多运行企业级应用系统所需的基础功能，包括动态服务请求、会话管理、身份验证、权限校验、数据缓存、日志跟踪、计划任务等，大

家可以把任擎当作 Node.js 使用，Node.js 支持的功能，任擎都支持。任擎采用 Node.js，一是看中了 Node.js 的高性能，二是使用 JavaScript 这一种语言就能进行前后端开发，实现全栈开发，不用学习多种语言，降低了学习门槛。

## 5.2 Node.js 的模块化

Node.js 是一个 JavaScript 运行环境，可以运行用 JavaScript 语言编写的脚本，我们在任擎服务器的 tasgine 目录内新建一个文本文件：test.txt，输入以下内容：

```
console.log('Hello TaskBuilder!');
```

保存，然后将文件扩展名改为.js，然后在 Windows 资源管理器的地址栏输入：cmd，然后回车，在打开的 DOS 窗口内输入以下命令：

```
tasgine test.js
```

回车，就会看到以下输出：

```
Hello TaskBuilder!
```

用 Node.js 执行 JavaScript 脚本就是这么简单，但我们在开发实际的应用系统时，代码肯定不会这么简单，一个应用系统会有大量的 JavaScript 代码，不可能将所有代码都放在一个文件里。为了方便管理和维护 js 代码，Node.js 提供了模块化管理的相关支持，可以将应用系统的代码拆分为很多模块分别保存到不同的文件中，在需要时，再添加这些模块的引用。

我们再来在 tasgine 目录内创建一个名为 hello.js 的文件，用记事本打开，输入以下内容：

```
//构造方法
var Hello = function() {
    //执行一些初始化操作
};

//定义对外公开的类名
```

```
module.exports = Hello;

//定义方法
Hello.prototype.test= function() {
    console.log('Hello TaskBuilder!');
}
```

保存，然后用记事本打开前面那个示例的 test.js 文件，将里面的内容改成下面的代码：

```
//添加 Hello 模块的引用
var Hello = require('hello');
//创建 Hello 类的实例对象
var hello = new Hello();
//调用 test 方法
hello.test();
```

保存，然后在 Windows 资源管理器的地址栏输入：cmd，然后回车，在打开的 DOS 窗口内输入以下命令：

```
tasgine test.js
```

可以看到输出的内容跟上面的例子一样。

上面是一个最简单的 js 模块，采用的 Node.js 的 CommonJS 模块化规范，实际的业务模块代码会比这个复杂，但整体结构就是这样。使用 TaskBuilder 开发的每一个后台服务都是一个 js 模块，也是采用的该规范，下面是一个任擎后台服务编译后的 js 代码示例：

```
//添加相关功能的引用
var util = require('util'); //添加 Node.js 系统模块 util 的引用，需要用它实现类的继承
var Service = require('Service'); //添加任擎后台服务根类的引用，tb 开发的所有后台服务都继承自该类
var Dao = require('Dao'); //添加任擎数据访问类的应用，以便进行数据库相关操作

//定义当前后台服务类的构造方法，所有后台服务必须定义该方法
var Delete = function() {
    Service.call(this); //执行后台服务根类的构造方法
```

```
};
```

```
//让当前后台服务类继承后台服务根类的属性和方法
```

```
util.inherits(Delete, Service);
```

```
//定义对外公开的类名
```

```
module.exports = Delete;
```

//定义后台服务的入口方法，在客户端请求该服务时，引擎会创建这个后台服务的实例对象，并调用其 process 方法

```
Delete.prototype.process = function(req, res) {
```

```
    //进行请求参数校验
```

```
    if(!req.id) {
```

```
        this.onLogicError(1, '请提供信息编号! ');
```

```
        return;
```

```
    }
```

```
    var self = this;
```

```
    //创建数据访问对象
```

```
    var dao = new Dao(this);
```

```
    //执行数据删除操作
```

```
    dao.delete('id='+req.id, function() {
```

```
        //结束当前后台服务，关闭数据库连接和网络连接
```

```
        self.end(res);
```

```
    });
```

```
};
```

从上面的代码可知，在一个js文件内，可以通过 require('文件路径') 的形式引入其他模块，如果是 Node.js 和引擎内置的模块，以及安装到 tasgine\node\_modules 目录里的 npm 模块，直接用 require('模块名称') 引用即可，如果是其他目录里的模块文件，则需要加上绝对路径或相对路径，例如，在 tasgine\app\project1\service 目录里的 service1.js 文件要引用 tasgine\app\project2\service 目录里的 service2.js，则可以用下面的代码进行引用：

```
let Service2 = require('../..project2/service2.js');
```



## 5.3 npm 包管理工具

### 5.3.1 npm 简介

npm (Node Package Manager) 是 Node.js 的默认包管理和分发工具，它允许开发者下载、安装、更新和卸载 Node.js 模块（包），从而简化了 JavaScript 项目的依赖管理过程。NPM 不仅是非官方的发布 Node 模块的标准，而且通过 Github 的收购，保证了其将永远免费使用。

自从 Node.js 面世后，世界各地的程序员基于 Node.js 开发了大量的功能，为了方便大家分享和使用这些功能，Node.js 社区推出了 npm 包管理器，js 程序员可以将自己开发的各种 js 模块（包）用 npm 工具打包后发布到 Node.js 统一的包管理平台，然后其他需要使用该 js 软件包的开发者，可以使用 `npm install` 命令下载安装这个软件包，然后在自己的 js 文件内用 `require('包名称')` 命令引用，从而可以利用 npm 包管理平台上的海量 js 软件包快速实现相关功能，不需要都自己从头开发。截至 2020 年 3 月 17 日，npm 为大约 1200 万开发人员提供了 130 万个软件包，这些开发人员每月下载这些软件包达 750 亿次，应该说，npm 是目前世界上最大的软件库之一，您能想到的各种 js 功能，npm 库里可能都已经存在了，大家有兴趣可以自己去官网搜索：<https://www.npmjs.com/>。

npm 的使用涵盖了从包的搜索、安装、使用到发布的整个流程，包括但不限于以下几点：

- **包的管理：**通过 `npm install` 命令可以安装项目中所需的包，而 `npm uninstall` 命令则用于卸载不需要的包。包的版本管理通过 `npm version` 命令实现，包括补丁版本、小版本和主版本的更新。
- **包的发布：**发布 npm 包需要先注册 npm 账号，然后在包的根目录运行 `npm publish` 命令。发布前需要确保包的 `package.json` 文件正确填写了包的名称、版本号、描述等信息，

并且遵循了 npm 的命名规范。

- **包的使用**: 在代码中, 通过 `require` 或 `import` 语句可以引入并使用 npm 包。此外, npm 包中的自定义组件可以通过在项目的配置文件中声明来使用。
- **包的构建**: 对于小程序等特定平台, 使用 npm 包可能需要进行额外的构建步骤。这通常涉及到在开发者工具中执行特定的构建命令, 以确保 npm 包能够正确集成到项目中。
- **包的依赖管理**: `package.json` 文件是 npm 包依赖管理的核心。它列出了项目依赖的所有包及其版本号, 确保了项目在不同环境下的可重复构建。

npm 的使用不仅限于 Node.js 开发, 它还广泛应用于前端开发中, 特别是在使用如 Webpack 等构建工具时, npm 包成为了开发过程中不可或缺的一部分。此外, npm 社区活跃, 拥有数量庞大的开源库和工具, 极大地丰富了开发者的选择。

### 5.3.2 在任擎中使用 npm 安装软件包

任擎服务器自带一些常用的 npm 包, 都安装在 `tasgine\node_modules` 目录内, 如果您的后台服务需要使用这些软件包, 则直接可以在您的后台服务里通过 `require('包名称')` 命令引用这些软件包, 如果您要使用的 npm 包在 `tasgine\node_modules` 目录内不存在, 则可以通过下面的操作进行安装:

(1) 打开 `tasgine` 文件夹, 在地址栏输入: `cmd`, 然后回车, 打开 DOS 窗口;

(2) 输入命令: `npm install 包名`, 然后按回车, 即可进行该软件包的安装;

等安装完毕后, 就可以在您的后台服务里引用该软件包了。

## 6、HTTP 协议

### 6.1 HTTP 简介

HTTP 协议（Hyper Text Transfer Protocol），全称为超文本传输协议，本质上是一种用于分布式、协助式的信息系统应用层协议，也是我们所熟悉的万维网的通讯基础。

HTTP 在 1989 年由欧洲的核子研究组织（CERN）发起，由万维网协会（W3C）与互联网工程组（IETF）讨论协调，多方共同参与制定才有了 HTTP1.1 作为第一个正式发表的标准。

HTTP 本质上是一种客户端（用户端）与服务器端（网站）进行数据的请求和应答的标准，相当于标准的数据交互必须满足这一统一标准，来方便各方对数据的解析与使用，而 HTTP 作为一种架设在 TCP 通信层之上的应用层协议，相当于对 TCP 传输的这类数据做了相关的规范性要求。例如我们访问某一个网站，那么客户端会向服务器发起一个 HTTP 的请求到指定的端口（通常为 80 端口），那么服务器在收到这个 HTTP 请求后，需要对这个请求产生应答，而应答的内容则是服务器上存储的 HTML 文件（也就是网页上看到的内容）和图像等数据，这些回复同样必须满足 HTTP 标准，否则客户端无法正常解析。

## 6.2 HTTP 工作原理

HTTP 协议定义了 Web 客户端如何从 Web 服务器请求 Web 页面，以及服务器如何把 Web 页面传送给客户端。HTTP 协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器以一个状态行作为响应，响应的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。

具体的过程如下：

### **(1) 客户端连接到 Web 服务器**

一个 HTTP 客户端，通常是浏览器，与 Web 服务器的 HTTP 端口（默认为 80）建立一个 TCP 套接字连接。

### **(2) 发送 HTTP 请求**

通过 TCP 套接字，客户端向 Web 服务器发送一个文本的请求报文，一个请求报文由请求行、请求头部、空行和请求数据 4 部分组成。

### (3) 服务器接受请求并返回 HTTP 响应

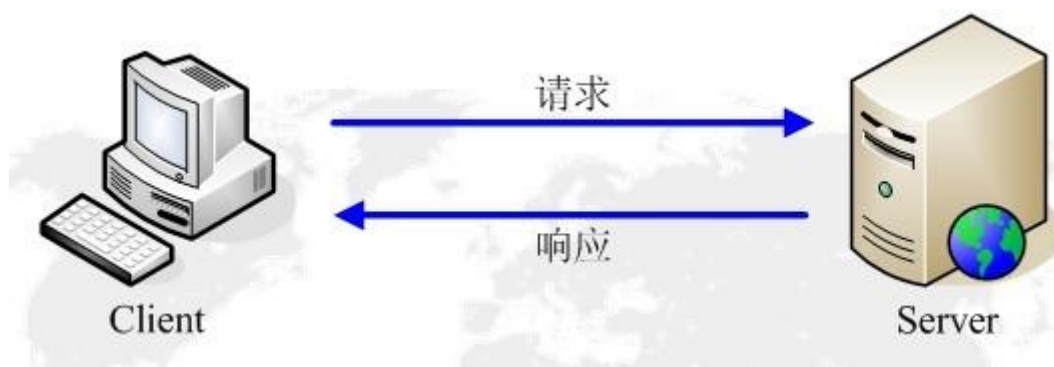
Web 服务器解析请求，定位请求资源。服务器将资源内容写到 TCP 套接字，由客户端读取。一个响应由状态行、响应头部、空行和响应数据 4 部分组成。

### (4) 释放连接 TCP 连接

若 connection 模式为 close，则服务器主动关闭 TCP 连接，客户端被动关闭连接，并释放 TCP 连接；若 connection 模式为 keepalive，则该连接会保持段时间，在该时间内可以继续接收请求。

### (5) 客户端浏览器解析 HTML 内容

客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的 HTML 文档和文档的字符集。客户端浏览器读取响应数据 HTML，根据 HTML 的语法对其进行格式化，并在浏览器窗口中显示。



## 6.3 HTTP 的特点

HTTP 协议具有以下特点：

### (1) 基于请求-响应的模式

HTTP 协议规定，请求从客户端发出，最后服务器端响应该请求并返回。换句话说，肯定是先从客户端开始建立通信的，服务器端在没有接收到请求之前不会发送响应。

### (2) 无状态保存

HTTP 是一种不保存状态，即无状态(stateless)协议。HTTP 协议自身不对请求和响应之间的通信状态进行保存。也就是说在 HTTP 这个级别，协议对于发送过的请求或响应都不做持久化处理。使用 HTTP 协议，每当有新的请求发送时，就会有对应的新响应产生。协议本身并不保留之前一切的请求或响应报文的信息，这是为了更快地处理大量请求，确保协议的可伸缩性，而特意把 HTTP 协议设计成如此简单的。可是，随着 Web 的不断发展，因无状态而导致业务处理变得棘手的情况增多了。比如，用户登录到一家购物网站，即使他跳转到该站的其他页面后，也需要能继续保持登录状态。针对这个实例，网站为了能够掌握是谁送出的请求，需要保存用户的状态。HTTP/1.1 虽然是无状态协议，但为了实现期望的保持状态功能，于是引入了 Cookie 技术。有了 Cookie 再用 HTTP 协议通信，就可以管理状态了。有关 Cookie 的详细内容请自行查看相关文档。

### (3) 无连接

无连接的含义是限制每次连接只处理一个请求，服务器处理完客户端的请求，并收到客户端的应答后，即断开连接。采用这种方式可以节省传输时间，并且可以提高并发性能，不能和每个用户建立长久的连接，请求一次响应一次，服务端和客户端就中断了。

## 6.4 常用 HTTP 方法

HTTP/1.1 协议中共定义了八种方法（也叫“动作”）来以不同方式操作指定的资源：

### 6.4.1 GET

向指定的资源发出内容获取请求，使用 GET 方法应该只用在读取数据，而不应当被用于添加、修改或删除等操作中，一个 GET 方法无法传输复杂、大量的数据，另外一个原因是 GET 可能会被网络蜘蛛等随意访问。

### 6.4.2 HEAD

与 GET 方法一样，都是向服务器发出指定资源的请求，只不过服务器将不传回资源的正文部分，只返回头部信息。它的好处在于，使用这个方法可以在不必传输全部内容的情况下，就可以获取其中“关于该资源的信息”（元信息或称元数据）。

### 6.4.3 POST

向指定资源提交数据，请求服务器进行处理（例如提交表单或者上传文件），数据被包含在请求主体中，这个请求可能会创建新的资源或修改现有资源，或二者皆有。

### 6.4.4 PUT

向指定资源位置上传其最新内容。

### 6.4.5 DELETE

请求服务器删除 Request-URI 所标识的资源。

### 6.4.6 TRACE

回显服务器收到的请求，主要用于测试或诊断。

### 6.4.7 OPTIONS

这个方法可使服务器传回该资源所支持的所有 HTTP 请求方法。用 "\*" 来代替资源名称，向

Web 服务器发送 OPTIONS 请求，可以测试服务器功能是否正常运作。

## 6.4.8 CONNECT

HTTP/1.1 协议中预留给能够将连接改为管道方式的代理服务器。通常用于 SSL 加密服务器的链接（经由非加密的 HTTP 代理服务器）。

用 TaskBuilder 进行低代码开发时，主要用到了 GET 和 POST 方法，请求静态资源时用 GET 方法，例如 ttf 页面、js、css 和图片文件等；请求后台服务时用的 POST 方法，因为一般请求后台服务都需要提交相关参数，当参数比较多或格式比较复杂时，用 GET 方法实现不了。

## 6.5 HTTP 响应状态码

所有 HTTP 响应的第一行都是状态行，依次是当前 HTTP 版本号、3 位数字组成的状态代码、以及描述状态的短语，彼此由空格分隔。状态代码的第一个数字代表当前响应的类型：

- **1xx 消息**：请求已被服务器接收，继续处理
- **2xx 成功**：请求已成功被服务器接收、理解、并接受
- **3xx 重定向**：需要后续操作才能完成这一请求
- **4xx 请求错误**：请求含有词法错误或者无法被执行
- **5xx 服务器错误**：服务器在处理某个正确请求时发生意外错误

虽然 RFC2616 中已经推荐了描述状态的短语，例如"200 OK"，"404 Not Found"，但是 WEB 开发者仍然能够自行决定采用何种短语，用以显示本地化的状态描述或者自定义信息。

## 6.6 HTTPS 协议

HTTP 虽然使用极为广泛，但是却存在不小的安全缺陷，主要是其数据的明文传送和消息完整性检测的缺乏，而这两点恰好是网络支付、网络交易等新兴应用中安全方面最需要关注的。



对于 HTTP 的明文数据传输，攻击者最常用的攻击手法就是网络嗅探，试图从传输过程当中分析出敏感的数据，例如管理员对 Web 程序后台的登录过程等等，从而获取网站管理权限，进而渗透到整个服务器的权限。即使无法获取到后台登录信息，攻击者也可以从网络中获取普通用户的隐私信息，包括手机号码、身份证号码、信用卡号等重要资料，导致严重的安全事故。进行网络嗅探攻击非常简单，对攻击者的要求很低。使用网络发布的任意一款抓包工具，一个新手就有可能获取到大型网站的用户信息。

另外，HTTP 在传输客户端请求和服务端响应时，唯一的数据完整性检验就是在报文头部包含了本次传输数据的长度，而对内容是否被篡改不作确认。因此攻击者可以轻易的发动中间人攻击，修改客户端和服务端传输的数据，甚至在传输数据中插入恶意代码，导致客户端被引导至恶意网站被植入木马。

为了提高 HTTP 协议的安全性，Netscape 公司提出了更加安全的 HTTPS (Hypertext Transfer Protocol Secure) 协议，并成为业界标准，HTTPS 协议是由 HTTP 加上 TLS/SSL 协议构建的可进行加密传输、身份认证的网络协议，主要通过数字证书、加密算法、非对称密钥等技术完成互联网数据传输加密，实现互联网传输安全保护。设计目标主要有三个。

**(1) 数据保密性：**保证数据内容在传输的过程中不会被第三方查看。就像快递员传递包裹一样，都进行了封装，别人无法获知里面装了什么。

**(2) 数据完整性：**及时发现被第三方篡改的传输内容。就像快递员虽然不知道包裹里装了什么东西，但他有可能中途掉包，数据完整性就是指如果被掉包，我们能轻松发现并拒收。

**(3) 身份校验安全性：**保证数据到达用户期望的目的地。就像我们邮寄包裹时，虽然是一个封装好的未掉包的包裹，但必须确定这个包裹不会送错地方，通过身份校验来确保送对了地方。

任擎服务器支持 HTTPS 协议，具体如何使用请参考任擎服务器的使用说明。

## 7、URL 格式

### 7.1 URL 简介

在互联网上,每一个信息资源都有统一的且唯一的地址,该地址就叫 URL(Uniform Resource Locator, 统一资源定位器),它是互联网的统一资源定位标志,就是指网络地址。

本文档后续内容中会经常涉及到 URL,例如通过 URL 传递参数,所以在此先提前介绍一下。

### 7.2 URL 语法

URL 由 4 部分组成: 协议、主机、端口、路径, URL 的一般语法格式为:

`protocol :// hostname[:port] / path / [:parameters][?query]#fragment`

其中带方括号的为可选项

### 7.3 URL 格式说明

#### 7.3.1 protocol (协议)

用来指定使用的传输协议,下面是比较常用的几种协议:

- **file**: 资源是本地计算机上的文件。格式 `file:///`, 注意后边应是三个斜杠。
- **ftp**: 可以通过 FTP 协议访问的资源。格式 `FTP://`
- **http**: 可以通过 HTTP 访问的资源。格式 `HTTP://`
- **https**: 可以通过安全的 HTTPS 访问的资源。格式 `HTTPS://`
- **mailto**: 资源为电子邮件地址, 通过 SMTP 访问。格式 `mailto:`
- **MMS**: 支持 MMS 协议的流媒体资源。(代表软件: Windows Media Player) 格式 `MMS://`
- **thunder**: 支持 thunder (专用下载链接) 协议的 P2P 软件访问该资源。(代表软件: 迅雷)

格式 thunder://

### 7.3.2 hostname (主机名)

是指存放资源的服务器的域名系统(DNS)主机名或 IP 地址。

### 7.3.3 port (端口号)

数据格式为整数，可选，各种传输协议都有默认的端口号，如 http 的默认端口为 80。如果输入时省略，则使用默认端口号。有时候出于安全或其他考虑，可以在服务器上对端口进行重定义，即采用非标准端口号，此时，URL 中就不能省略端口号这一项。

### 7.3.4 path (路径)

由零或多个 “/” 符号隔开的字符串，一般用来表示主机上的一个目录或文件地址。

### 7.3.5 parameters (参数)

这是用于指定特殊参数的可选项，有服务器端程序自行解释。

### 7.3.6 query (查询)

可选，用于给动态网页（如使用 PHP/JSP/ASP/ASP.NET 等技术制作的网页或 TaskBuilder 开发的后台服务）传递参数，可有多参数，用 “&” 符号隔开，每个参数的名和值用 “=” 符号隔开。

### 7.3.7 fragment (信息片断)

字符串，用于指定网络资源中的片段。例如一个网页中有多个名词解释，可使用 fragment 直接定位到某一名词解释。

## 五、TaskBuilder 低代码开发初体验

### 1、示例应用简介

为了让大家快速了解 TaskBuilder 的相关功能，我们将带领大家一步一步实现一个简单的增删改查应用。

这个示例应用假定是为某个培训学校开发一个管理学员基本信息的应用，我们提供了一份用来测试的学员名单，下载地址：

<http://www.taskbuilder.org/app/website/doc/xyxx.xlsx>

内容如下图所示：

|    | A    | B  | C  | D        | E   | F  | G                  | H    | I   | J   | K     | L           |
|----|------|----|----|----------|-----|----|--------------------|------|-----|-----|-------|-------------|
| 1  | 培训年份 | 期数 | 班次 | 学号       | 姓名  | 性别 | 身份证号               | 文化程度 | 乡镇  | 行政村 | 是否贫困户 | 电话号码        |
| 2  | 2021 | 1期 | 1班 | 21010101 | 王艳  | 女  | 123456789012340001 | 小学   | 温柔乡 | 富贵村 | 否     | 13512343001 |
| 3  | 2021 | 1期 | 1班 | 21010102 | 武文梅 | 女  | 123456789012340002 | 小学   | 温柔乡 | 富贵村 | 是     | 13512343002 |
| 4  | 2021 | 1期 | 1班 | 21010103 | 樊伶  | 女  | 123456789012340003 | 大学   | 温柔乡 | 富贵村 | 否     | 13512343003 |
| 5  | 2021 | 1期 | 2班 | 21010201 | 李百兰 | 女  | 123456789012340004 | 小学   | 温柔乡 | 富贵村 | 否     | 13512343004 |
| 6  | 2021 | 1期 | 2班 | 21010202 | 樊彩梅 | 女  | 123456789012340005 | 初中   | 温柔乡 | 富贵村 | 否     | 13512343005 |
| 7  | 2021 | 1期 | 2班 | 21010203 | 杜平  | 女  | 123456789012340006 | 初中   | 温柔乡 | 富贵村 | 否     | 13512343006 |
| 8  | 2021 | 1期 | 3班 | 21010301 | 薛林平 | 女  | 123456789012340007 | 初中   | 温柔乡 | 富贵村 | 否     | 13512343007 |
| 9  | 2021 | 1期 | 3班 | 21010302 | 李旭英 | 女  | 123456789012340008 | 初中   | 温柔乡 | 富贵村 | 是     | 13512343008 |
| 10 | 2021 | 1期 | 3班 | 21010303 | 李晓丽 | 女  | 123456789012340009 | 初中   | 温柔乡 | 富贵村 | 否     | 13512343009 |
| 11 | 2021 | 2期 | 1班 | 21020101 | 李贵  | 男  | 123456789012340010 | 初中   | 国强镇 | 振兴村 | 是     | 13512343010 |
| 12 | 2021 | 2期 | 1班 | 21020102 | 姚平勤 | 女  | 123456789012340011 | 初中   | 国强镇 | 振兴村 | 否     | 13512343011 |
| 13 | 2021 | 2期 | 1班 | 21020103 | 刘连新 | 女  | 123456789012340012 | 高中   | 国强镇 | 振兴村 | 否     | 13512343012 |
| 14 | 2021 | 2期 | 1班 | 21020104 | 刘春兰 | 女  | 123456789012340013 | 初中   | 国强镇 | 振兴村 | 是     | 13512343013 |
| 15 | 2021 | 2期 | 2班 | 21020201 | 薛淑平 | 女  | 123456789012340014 | 初中   | 国强镇 | 振兴村 | 否     | 13512343014 |
| 16 | 2021 | 2期 | 2班 | 21020202 | 薛耀东 | 男  | 123456789012340015 | 初中   | 国强镇 | 振兴村 | 否     | 13512343015 |
| 17 | 2021 | 2期 | 2班 | 21020203 | 赵利利 | 女  | 123456789012340016 | 初中   | 国强镇 | 振兴村 | 否     | 13512343016 |
| 18 | 2021 | 2期 | 3班 | 21020301 | 武红梅 | 女  | 123456789012340017 | 初中   | 国强镇 | 振兴村 | 是     | 13512343017 |
| 19 | 2021 | 2期 | 3班 | 21020302 | 任小凤 | 女  | 123456789012340018 | 小学   | 国强镇 | 振兴村 | 否     | 13512343018 |
| 20 | 2021 | 2期 | 3班 | 21020303 | 严利丽 | 女  | 123456789012340019 | 初中   | 国强镇 | 振兴村 | 否     | 13512343019 |
| 21 | 2021 | 3期 | 1班 | 21030101 | 张艳艳 | 女  | 123456789012340020 | 初中   | 银河乡 | 地球村 | 是     | 13512343020 |
| 22 | 2021 | 3期 | 1班 | 21030102 | 秦艳梅 | 女  | 123456789012340021 | 初中   | 银河乡 | 地球村 | 否     | 13512343021 |
| 23 | 2021 | 3期 | 1班 | 21030103 | 王峰  | 女  | 123456789012340022 | 小学   | 银河乡 | 地球村 | 否     | 13512343022 |
| 24 | 2021 | 3期 | 2班 | 21030201 | 张瑞云 | 女  | 123456789012340023 | 小学   | 银河乡 | 地球村 | 否     | 13512343023 |
| 25 | 2021 | 3期 | 2班 | 21030202 | 高乃顺 | 男  | 123456789012340024 | 高中   | 银河乡 | 地球村 | 是     | 13512343024 |
| 26 | 2021 | 3期 | 2班 | 21030203 | 李平  | 女  | 123456789012340025 | 小学   | 银河乡 | 地球村 | 否     | 13512343025 |
| 27 | 2021 | 3期 | 2班 | 21030204 | 霍星宇 | 女  | 123456789012340026 | 大专   | 银河乡 | 地球村 | 否     | 13512343026 |
| 28 | 2021 | 3期 | 3班 | 21030301 | 李荣梅 | 女  | 123456789012340027 | 初中   | 银河乡 | 地球村 | 否     | 13512343027 |
| 29 | 2021 | 3期 | 3班 | 21030302 | 王维平 | 女  | 123456789012340028 | 初中   | 银河乡 | 地球村 | 是     | 13512343028 |
| 30 | 2021 | 3期 | 3班 | 21030303 | 王利建 | 女  | 123456789012340029 | 初中   | 银河乡 | 地球村 | 否     | 13512343029 |

**说明：**上述 Excel 文件里的数据只是为了演示 TaskBuilder 的功能而随意编写的，并非真实数据，如果姓名有雷同，纯属巧合。

我们要基于以上 Excel 文件的格式和内容，实现以下功能：

- 1、实现学员基本信息的列表查看，可以按姓名、学号、性别、文化程度等条件查询；
- 2、可以手动添加学员信息；
- 3、可以修改指定学员的信息；
- 4、可以删除指定学员的信息；

TaskBuilder 提供了强大的向导和便捷的功能，可以快速实现上述功能，下面就让我们开始吧。

## 2、创建学员管理项目

打开 TaskBuilder，点击左侧项目资源管理器顶部工具栏上的加号按钮：



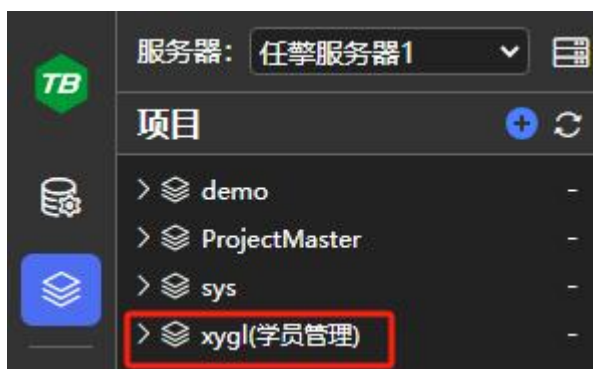
在弹出的对话框中输入以下信息：

**项目代码：**xygl（只能是字母、数字或下划线）

**项目说明：**学员管理



点击确定按钮，即可完成项目的创建，在 TaskBuilder 的项目资源管理器内就可以看到这个新建的项目了，如下图所示：



点击该项目节点，可以展开看到该项目下面有 5 个子节点，如下图所示：



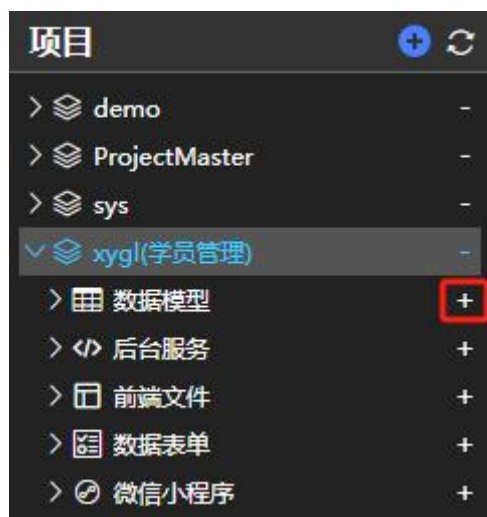
后续就可以在此继续添加、查看、修改和删除该项目的数据模型、后台服务、前端页面、数据表单和微信小程序等内容。

### 3、创建学员信息数据模型

使用 TaskBuilder 开发应用的第一步就是先创建数据模型，数据模型是用来存储应用数据结构的文件，可以定义各项业务数据有哪些字段，每个字段的数据类型、长度、是否为空等，TaskBuilder 提供了一个可视化的数据模型设计器，可以不借助任何其他第三方工具，完成应用数据模型的设计。

在 TaskBuilder 的项目资源管理器内，点击学员管理项目节点（xygl）里的“数据模型”子节点右侧的加号：





在弹出的对话框中选择“新建表”，然后点下一步。



打开 **新建数据模型** 对话框，如下图所示：

新建数据模型

模型路径: /app/xygl/model

模型名称: xyxx 只能是字母、数字和下划线，且以字母开头

模型备注: 学员信息

数据源: 任擎默认数据库

创建常用字段:

| <input type="checkbox"/>            | 字段名          | 数据类型     | 长度 | 字段备注  | 其他            |
|-------------------------------------|--------------|----------|----|-------|---------------|
| <input checked="" type="checkbox"/> | id           | INT      | 11 | 编号    | 主键, 自动增长      |
| <input type="checkbox"/>            | code         | VARCHAR  | 20 | 编码    |               |
| <input type="checkbox"/>            | name         | VARCHAR  | 20 | 名称    |               |
| <input type="checkbox"/>            | parent_id    | INT      | 11 | 父级编号  | 默认为0          |
| <input type="checkbox"/>            | creator_id   | INT      | 11 | 创建人编号 |               |
| <input type="checkbox"/>            | create_time  | DATETIME |    | 创建时间  |               |
| <input type="checkbox"/>            | auditor_id   | INT      | 11 | 审核人编号 |               |
| <input type="checkbox"/>            | audit_time   | DATETIME |    | 审核时间  |               |
| <input type="checkbox"/>            | audit_result | VARCHAR  | 20 | 审核结果  | Y: 同意, N: 不同意 |

确定 取消

在该对话框中输入以下信息：

**模型名称：**xyxx（只能是字母、数字或下划线）

**模型备注：**学员信息

并在下面的常用字段列表里选中 id 字段。

然后点确定按钮，会打开 **数据模型设计器**，然后点击工具栏上的“添加字段”按钮可以添加新字段信息的设置行，参考学员信息 Excel 文件里的内容，为每一列添加一个对应的字段，如下图所示：

| <input type="checkbox"/>            | 字段名   | 字段类型 | 长度 | 精度 | 非空                                  | 主键                                  | 自增长                                 | 默认值 | 备注    |
|-------------------------------------|-------|------|----|----|-------------------------------------|-------------------------------------|-------------------------------------|-----|-------|
| <input checked="" type="checkbox"/> | id    | 整数   |    |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |     | 学员编号  |
| <input type="checkbox"/>            | pxnf  | 整数   |    |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 培训年份  |
| <input type="checkbox"/>            | qs    | 字符串  | 10 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 期数    |
| <input type="checkbox"/>            | bc    | 字符串  | 10 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 班次    |
| <input type="checkbox"/>            | xh    | 字符串  | 10 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 学号    |
| <input type="checkbox"/>            | xm    | 字符串  | 10 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 姓名    |
| <input type="checkbox"/>            | xb    | 字符串  | 2  |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 性别    |
| <input type="checkbox"/>            | sfzh  | 字符串  | 24 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 身份证号  |
| <input type="checkbox"/>            | whcd  | 字符串  | 10 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 文化程度  |
| <input type="checkbox"/>            | xz    | 字符串  | 20 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 乡镇    |
| <input type="checkbox"/>            | xzc   | 字符串  | 20 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 行政村   |
| <input type="checkbox"/>            | sfpkh | 字符串  | 2  |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 是否贫困户 |
| <input type="checkbox"/>            | dhhm  | 字符串  | 20 |    | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 电话号码  |

其中，id 字段是为了后续开发相关功能时方便各个页面之间用该 id 作为每条学员信息的唯一标识，以便关联查找对应的信息，该字段的数据类型必须设置为整数，且必须是主键和自动增长，也不能为空。其他字段可以根据对应的信息内容选择适当的数据类型和长度。

每个字段必须设置字段名和字段的数据类型，如果是字符串，还需要设置好适当的长度。并且备注也必须设置一下，且最好用汉字，后续页面里会自动用备注作为对应输入项的标题。

字段都设置好后，点击工具栏上的保存按钮，即可完成数据模型的创建，TaskBuilder 会自动在数据库中创建该数据模型对应的表，不需要再手动创建。

**注意：**



(1) 考虑到大家的英文水平参差不齐，这个演示项目的字段名用的拼音简称，没有采用英文单词，大家可以通过备注了解字段的具体含义。大家在开发自己的应用系统时，字段名具体用拼音还是英文字母由您自己决定，字段名也能用汉字，但不推荐，推荐用英文单词。

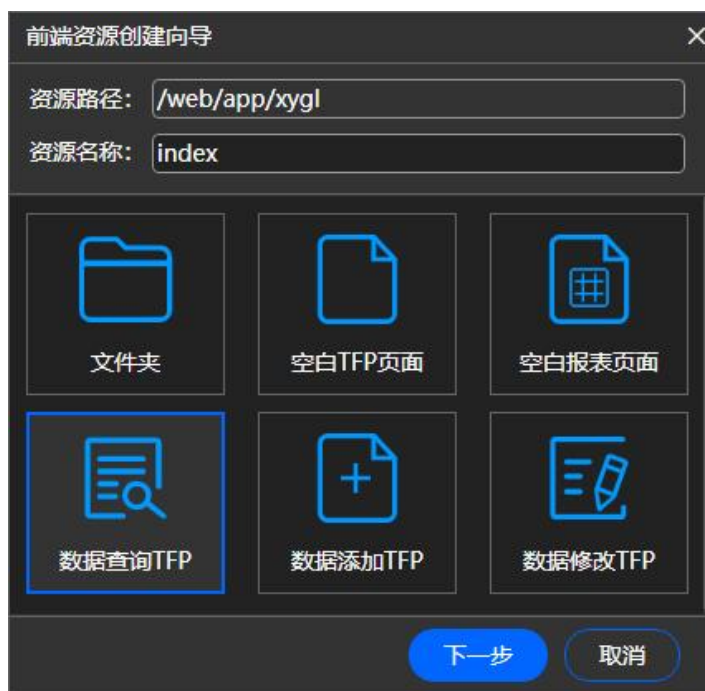
(2) 字段名建议都用小写字母，因为 SQL Server 默认会将所有的标识符转换为小写字母后再进行处理，也就是说如果在数据模型里字段名用大写字母，则根据该数据模型创建的页面里，字段对应的数据列或输入项的数据绑定表达式里也会用大写字母的字段名，但查询数据时，数据库返回的字段名却都是小写的，就会造成数据无法正确绑定，因为 JavaScript 是区分大小写的。

## 4、创建学员信息列表页面

在 TaskBuilder 的项目资源管理器内，点击学员管理项目节点 (xygl) 里的“前端文件”子节点右侧的加号：



会弹出如下所示的 **前端资源创建向导** 的对话框：



在该对话框中选择“数据查询 TFP”，资源名称会自动设置为：index（这里建议不要修改，否则可能会导致后续创建的功能无法正常使用），然后点击“下一步”按钮，打开如下图所示的“创建数据查询 TFP”向导对话框：



创建数据查询TFP

第一步：设置基本信息

保存路径：/web/app/xygl/index.tfp 数据模型：xygl/model/xyxx.tdm 选择...

页面标题：学员信息管理 页面类型：普通页面 窗口大小：宽 高

终端类型：电脑端 背景颜色：浅色 定位方式：浮动定位

查询设置：☒ 启用翻页 每页最多显示 20 条记录 最多查询 条记录 ☐ 去掉重复项

其他设置：☒ 显示添加按钮 添加页面文件名：add.tfp 添加页面类型：抽屉页

☒ 显示修改按钮 修改页面文件名：update.tfp 修改页面类型：抽屉页

☒ 显示删除按钮

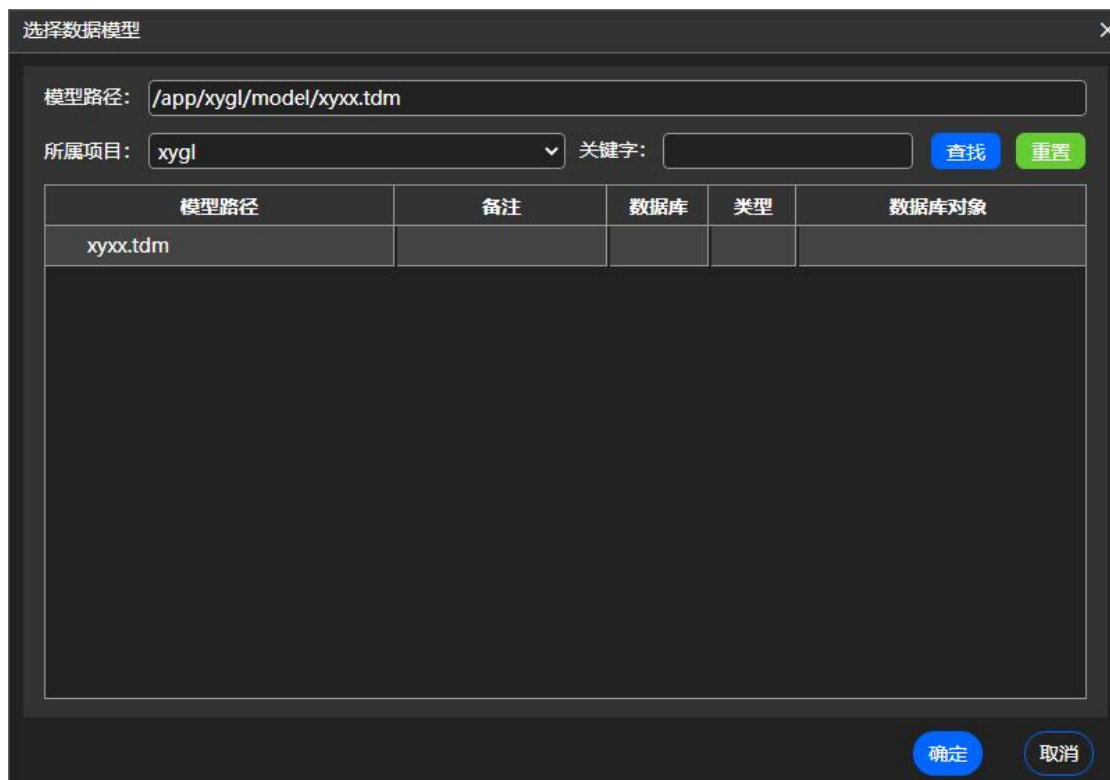
☐ 查询条件显示在页面左侧（仅支持PC端）

数据查询服务：☒ 自动创建 ☐ 使用已有服务 响应参数名：data 选择...

数据删除服务：☒ 自动创建 ☐ 使用已有服务 选择...

下一步 取消

创建数据列表页面的第一步就是先设置页面基本信息，点击数据模型输入框右侧的“选择”按钮，打开 **选择数据模型** 对话框，如下图所示：



选择数据模型

模型路径：/app/xygl/model/xyxx.tdm

所属项目：xygl 关键字： 查找 重置

| 模型路径     | 备注 | 数据库 | 类型 | 数据库对象 |
|----------|----|-----|----|-------|
| xyxx.tdm |    |     |    |       |

确定 取消

选择前面创建的 xyxx 这个数据模型，然后点确定，然后输入页面标题“学员信息管理”，其他选项使用默认值即可，不需要设置，然后点击“下一步”按钮，显示如下所示界面：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段

添加关联字段

| <input checked="" type="checkbox"/> | 字段名称  | 字段备注  | 字段格式(双击编辑) | 关联数据字典(双击选择) | 字段别名 |
|-------------------------------------|-------|-------|------------|--------------|------|
| <input checked="" type="checkbox"/> | id    | 编号    | id         |              |      |
| <input checked="" type="checkbox"/> | pxnf  | 培训年份  | pxnf       |              |      |
| <input checked="" type="checkbox"/> | qs    | 期数    | qs         |              |      |
| <input checked="" type="checkbox"/> | bc    | 班次    | bc         |              |      |
| <input checked="" type="checkbox"/> | xm    | 姓名    | xm         |              |      |
| <input checked="" type="checkbox"/> | xb    | 性别    | xb         |              |      |
| <input checked="" type="checkbox"/> | sfzh  | 身份证号  | sfzh       |              |      |
| <input checked="" type="checkbox"/> | whcd  | 文化程度  | whcd       |              |      |
| <input checked="" type="checkbox"/> | xz    | 乡镇    | xz         |              |      |
| <input checked="" type="checkbox"/> | xzc   | 行政村   | xzc        |              |      |
| <input checked="" type="checkbox"/> | sfpkh | 是否贫困户 | sfpkh      |              |      |
| <input checked="" type="checkbox"/> | dhhm  | 电话号码  | dhhm       |              |      |

上一步

下一步

取消

在这一步，需要选择具体要从数据库中查询哪些字段，并可以设置字段格式和别名，点击字段列表顶部列标题最左侧的复选框，可以全选或取消选中全部字段，这里我们选中全部字段，其他参数不用设置，然后点击下一步按钮，显示如下所示界面：

创建数据查询TFP

第三步：设置显示格式

+

| 列名    | 显示格式(js表达式) | 对齐方式 | 宽度  | 文字颜色 | 背景颜色 | 加粗                       | 求和                       |
|-------|-------------|------|-----|------|------|--------------------------|--------------------------|
| 培训年份  | {pxnf}      | 居中对齐 |     |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 期数    | {qs}        | 居中对齐 | 60  |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 班次    | {bc}        | 居中对齐 | 60  |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 姓名    | {xm}        | 居中对齐 |     |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 性别    | {xb}        | 居中对齐 | 60  |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 身份证号  | {sfzh}      | 居中对齐 | 150 |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 文化程度  | {whcd}      | 居中对齐 |     |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 乡镇    | {xz}        | 居中对齐 |     |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 行政村   | {xzc}       | 居中对齐 |     |      |      | <input type="checkbox"/> | <input type="checkbox"/> |
| 是否贫困户 | {sfpkh}     | 居中对齐 |     |      |      | <input type="checkbox"/> | <input type="checkbox"/> |

上一步

下一步

取消

在这一步，需要设置在列表页面里具体要显示哪些字段，以及每个字段的显示格式、对齐方式、宽度、文本颜色、背景颜色、是否加粗、是否求和等，因为学员编号字段是系统自动设置的，是为了关联查询用的，没有实际业务含义，不需要在页面里显示，所以可以在此把这个字段删掉，其他字段保留，并参考上图设置一下宽度（默认为 80 像素），然后点击“下一步”按钮，会显示如下内容：

创建数据查询TFP

第四步：设置动态查询参数

查询字段：  查询条件：  +

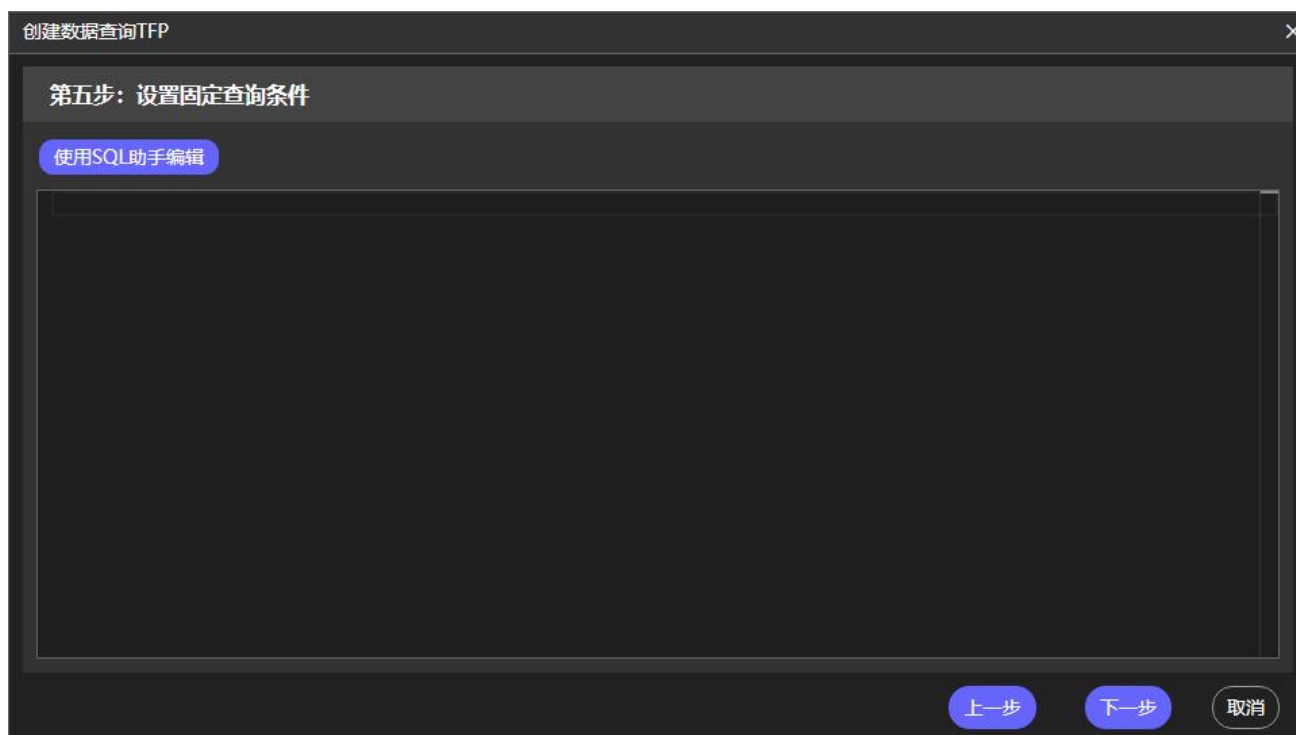
| 参数名称 | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)            | 默认值 | 数据字典(双击选择) | 可选值         |
|------|------|-------|----|------------------------|-----|------------|-------------|
| pxnf | 培训年份 | 单行文本  |    | pxnf={req.pxnf}        |     |            |             |
| qs   | 期数   | 单行文本  |    | qs={req.qs}            |     |            |             |
| xm   | 姓名   | 单行文本  |    | xm like {%req.xm%}     |     |            |             |
| sfzh | 身份证号 | 单行文本  |    | sfzh like {%req.sfzh%} |     |            |             |
| whcd | 文化程度 | 下拉列表  |    | whcd={req.whcd}        |     |            | 小学,初中,高中,大专 |

上一步 下一步 取消

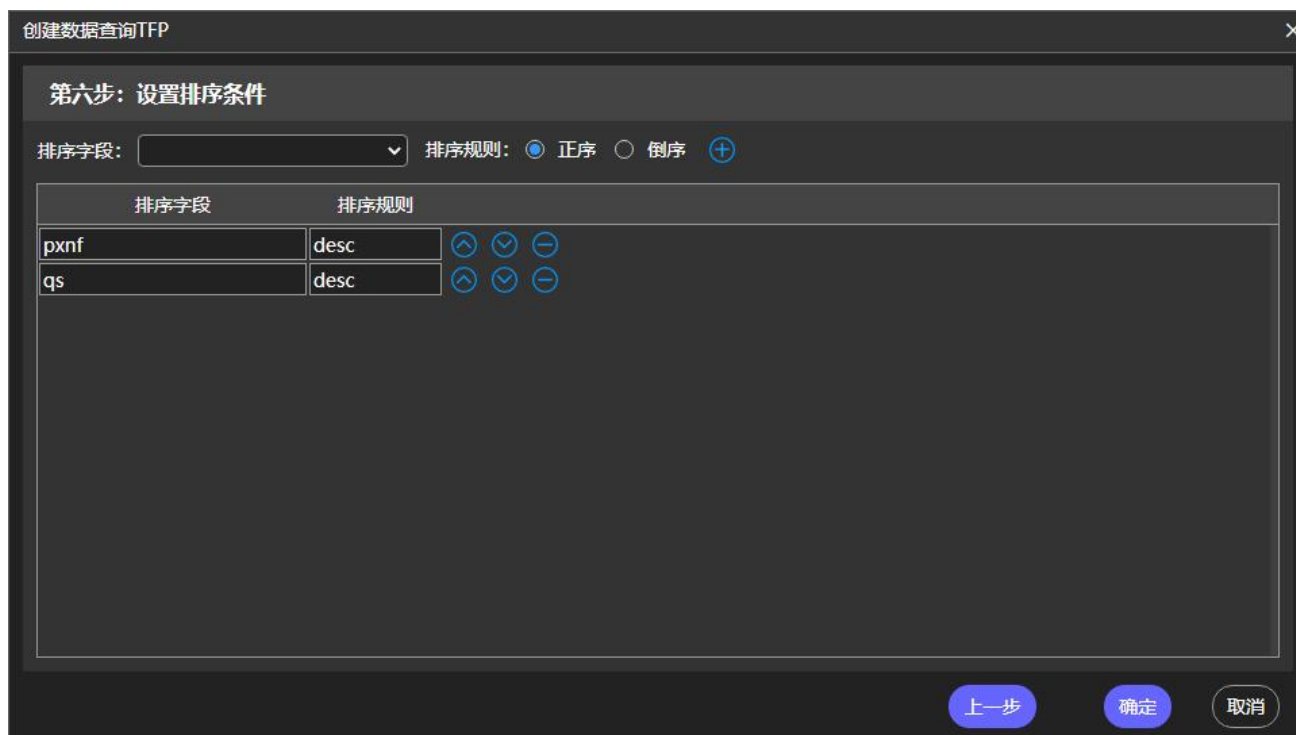
在这一步，可以设置列表页面里需要提供的动态查询条件，可以添加多个查询条件，具体的操作如下：

- 1、添加查询选项：**先在顶部查询字段列表里选择需要作为查询条件的字段，然后在右侧的查询条件列表里选择查询条件的类型，然后点击蓝色的加号图标，即可完成查询选项的添加。
- 2、设置查询选项：**添加完查询参数后，还需要设置参数说明、查询输入项的类型、宽度等参数，如果输入项选的下拉列表、单选框、复选框这几个类型，则还需要在后面的可选值里设置具体有哪几个值可选，多个值之间使用英文逗号分隔。

查询选项设置好之后，点击“下一步”按钮，可以看到如下界面：

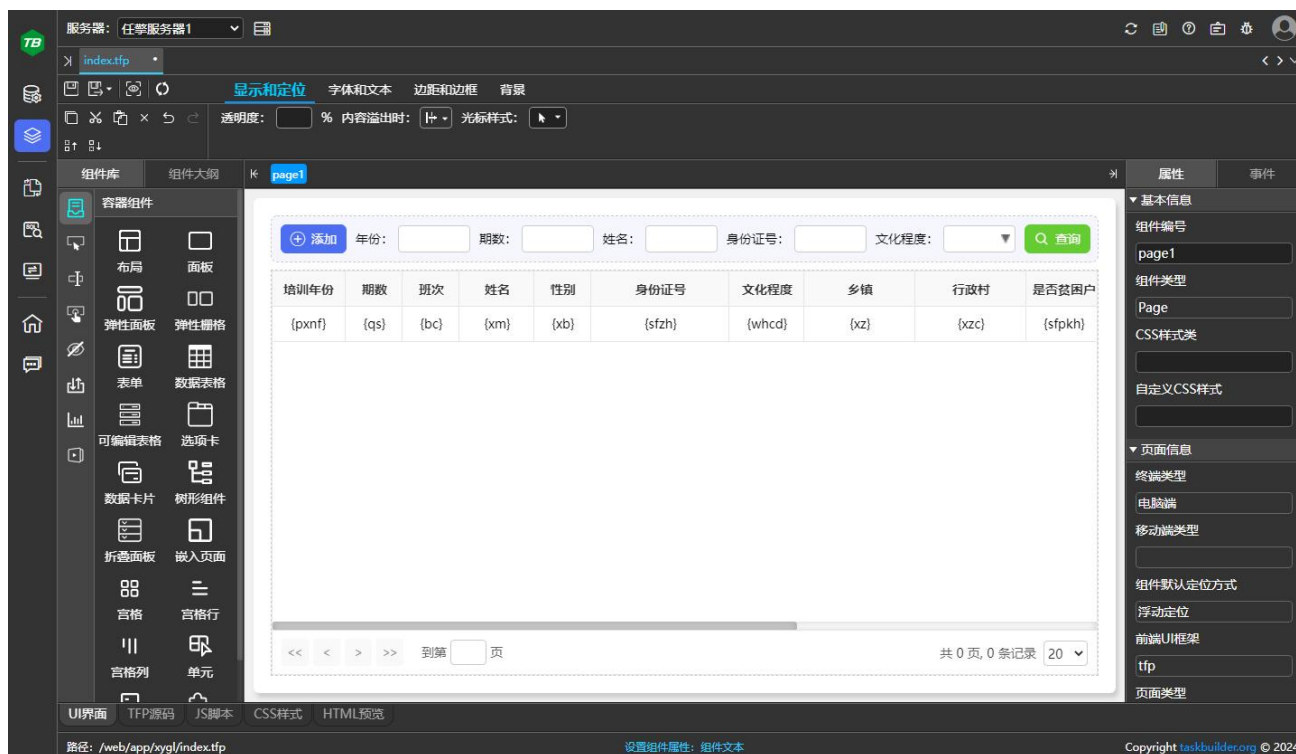


这一步用来设置默认的、固定的查询条件，例如只查某个状态或者某个用户的数据，本示例应用不需要设置任何默认查询条件，直接点击下一步即可，可以看到如下界面：



在此可以设置数据排序的规则，在上面的排序字段列表里选中要排序的字段，然后在排序规则里选择是正序排列还是倒序排列，然后点右侧的加号按钮，即可添加相应的排序规则，本示例选择了培训年份和期数这两个字段，且排序规则都设置为倒序。

然后点确定按钮，即可完成数据列表页的创建，TaskBuilder 会自动打开 **前端页面设计器** 显示这个新创建的 tfp 页面，界面如下图所示：



这个“学员信息列表”页面创建好之后，先不用管它，请继续查看后续的文档，跟我们一起创建“添加学员信息”页面。

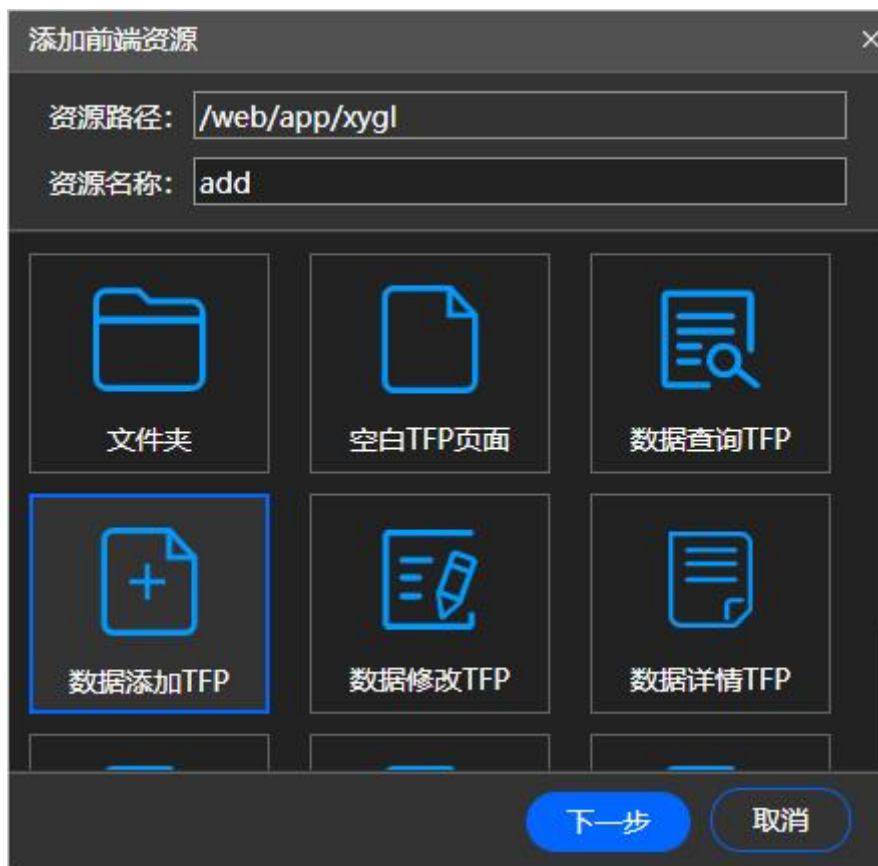
## 5、创建学员信息添加页面

在 TaskBuilder 的项目资源管理器内，点击学员管理项目节点（xygl）里的“前端文件”子节点右侧的加号：



会弹出如下所示的“添加前端资源”的对话框：





在“资源名称”输入框内输入：add，然后在下面的资源类型列表里点击“数据添加TFP”按钮，然后点击“下一步”按钮，打开如下图所示的“创建数据添加TFP”向导对话框：

创建数据添加TFP

保存路径: /web/app/xygl/add.tfp 数据模型: xygl/model/xyxx.tdm 选择...

页面标题: 添加学员信息 页面类型: 抽屉页 窗口大小: 宽 高

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧 每行显示输入项数量: 3 仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务 选择...

| <input checked="" type="checkbox"/> | 字段    | 输入项名称 | 输入项类型 | 宽度 | 占用列数 | 必填                                  | 可选值        | 数据字典(双击选择) | 默认值 |
|-------------------------------------|-------|-------|-------|----|------|-------------------------------------|------------|------------|-----|
| <input checked="" type="checkbox"/> | xb    | 性别    | 单选框   |    |      | <input checked="" type="checkbox"/> | 男,女        |            | 男   |
| <input checked="" type="checkbox"/> | sfzh  | 身份证号  | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | whcd  | 文化程度  | 下拉列表  |    |      | <input checked="" type="checkbox"/> | 小学,初中,高中,大 |            |     |
| <input checked="" type="checkbox"/> | xz    | 乡镇    | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | xzc   | 行政村   | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | sfpkh | 是否贫困户 | 开关    |    |      | <input checked="" type="checkbox"/> | 是,否        |            | 否   |
| <input checked="" type="checkbox"/> | dhhm  | 电话号码  | 单行输入框 |    |      | <input checked="" type="checkbox"/> |            |            |     |
| <input checked="" type="checkbox"/> | bz    | 备注    | 多行输入框 |    | 3    | <input type="checkbox"/>            |            |            |     |

确定 取消

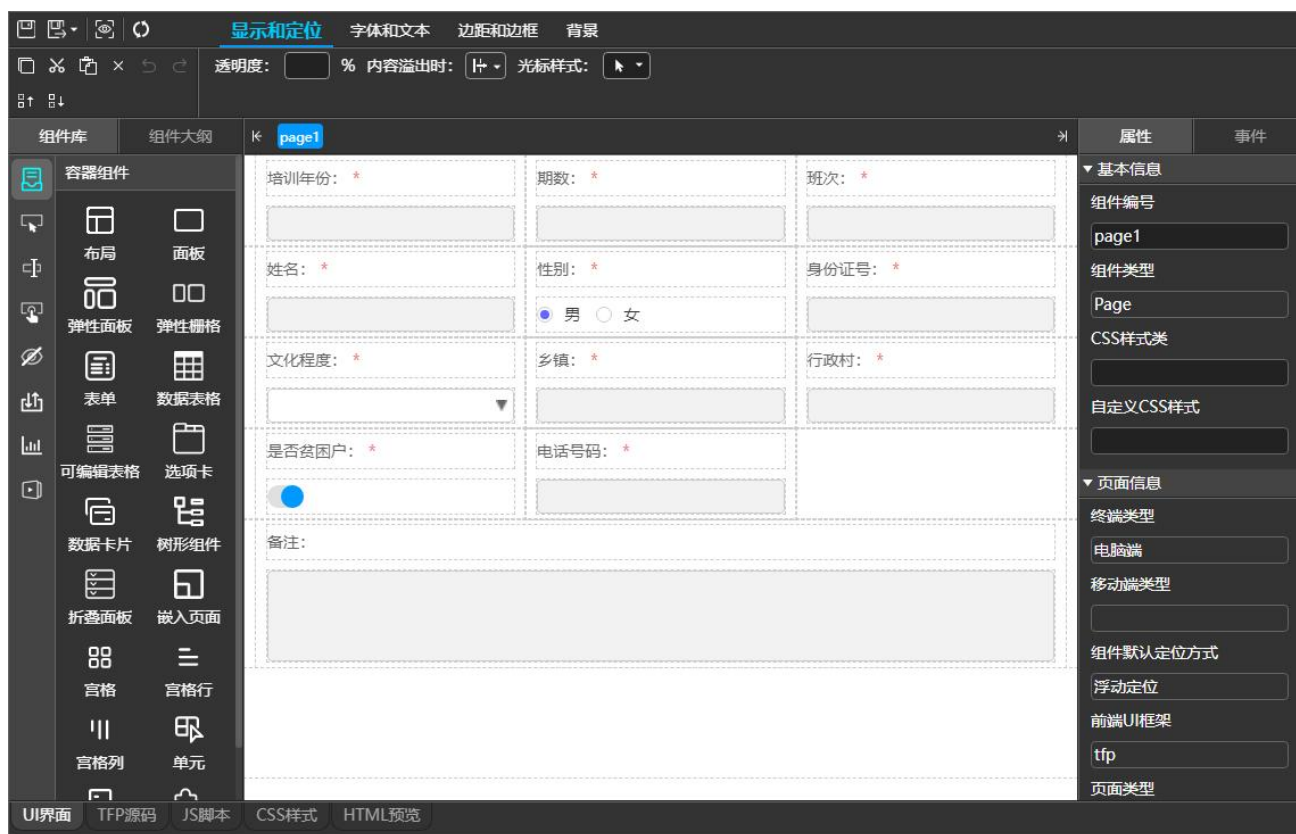


在该对话框内，按以下步骤操作：

- (1) 先点击“数据模型”输入框右侧的“选择...”按钮，在打开的对话框里选择前面创建的“xyxx.tdm”这个数据模型，点确定按钮。
- (2) 在“页面标题”输入框内输入“添加学员信息”，其他选项默认即可，不用设置。
- (3) 点击字段列表里标题栏最左侧的复选框，全选所有字段，并按下表所示设置各个字段的相关参数：

| 字段    | 输入项名称 | 输入项类型 | 占用<br>列数 | 必填 | 可选值            | 默认值 |
|-------|-------|-------|----------|----|----------------|-----|
| pxnf  | 培训年份  | 单行输入框 |          | 是  |                |     |
| qs    | 期数    | 单行输入框 |          | 是  |                |     |
| bc    | 班次    | 单行输入框 |          | 是  |                |     |
| xm    | 姓名    | 单行输入框 |          | 是  |                |     |
| xb    | 性别    | 单选框   |          | 是  | 男,女            | 男   |
| sfzh  | 身份证号  | 单行输入框 |          | 是  |                |     |
| whcd  | 文化程度  | 下拉列表  |          | 是  | 小学,初中,高中,大专,本科 |     |
| xz    | 乡镇    | 单行输入框 |          | 是  |                |     |
| xzc   | 行政村   | 单行输入框 |          | 是  |                |     |
| sfpkh | 是否贫困户 | 开关    |          | 是  | 是,否            | 否   |
| dghm  | 电话号码  | 单行输入框 |          | 是  |                |     |
| bz    | 备注    | 多行输入框 | 3        | 否  |                |     |

然后点击“确定”按钮，即可完成学员信息添加页面的创建，TaskBuilder 会根据我们前面选择的字段及相关设置，自动创建对应的 tfp 页面以及输入项，如下图所示：



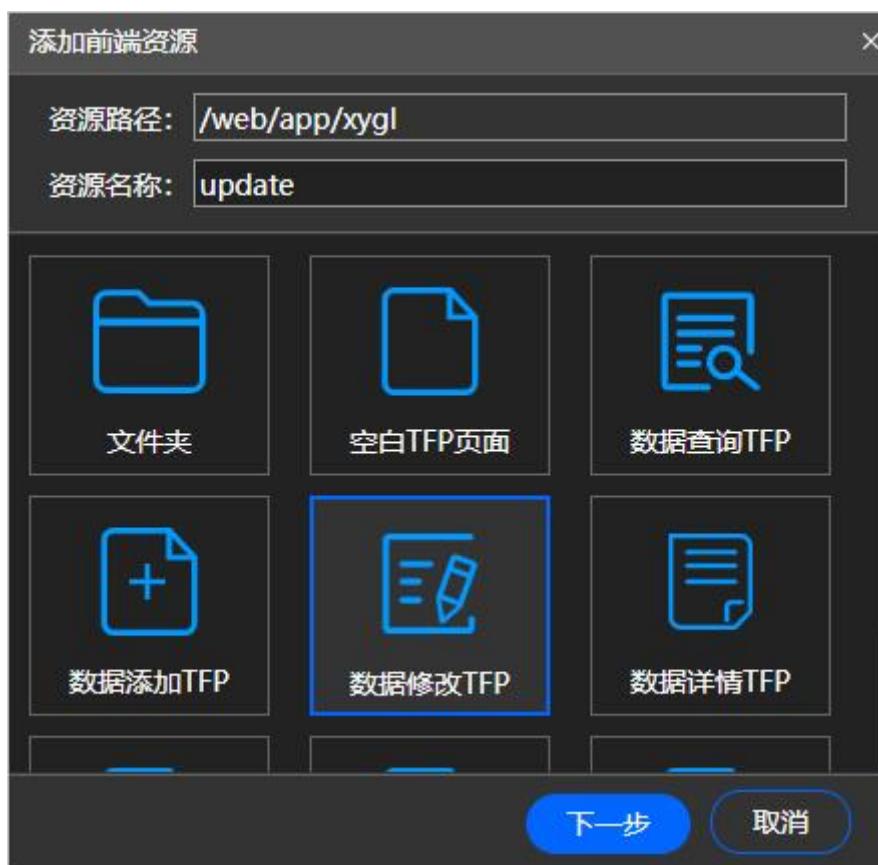
学员信息添加页面创建完后，先不用管它，请继续查看后面的文档。

## 6、创建学员信息修改页面

在 TaskBuilder 的项目资源管理器内，点击学员管理项目节点（xygl）里的“前端文件”子节点右侧的加号：



会弹出如下所示的“添加前端资源”的对话框：



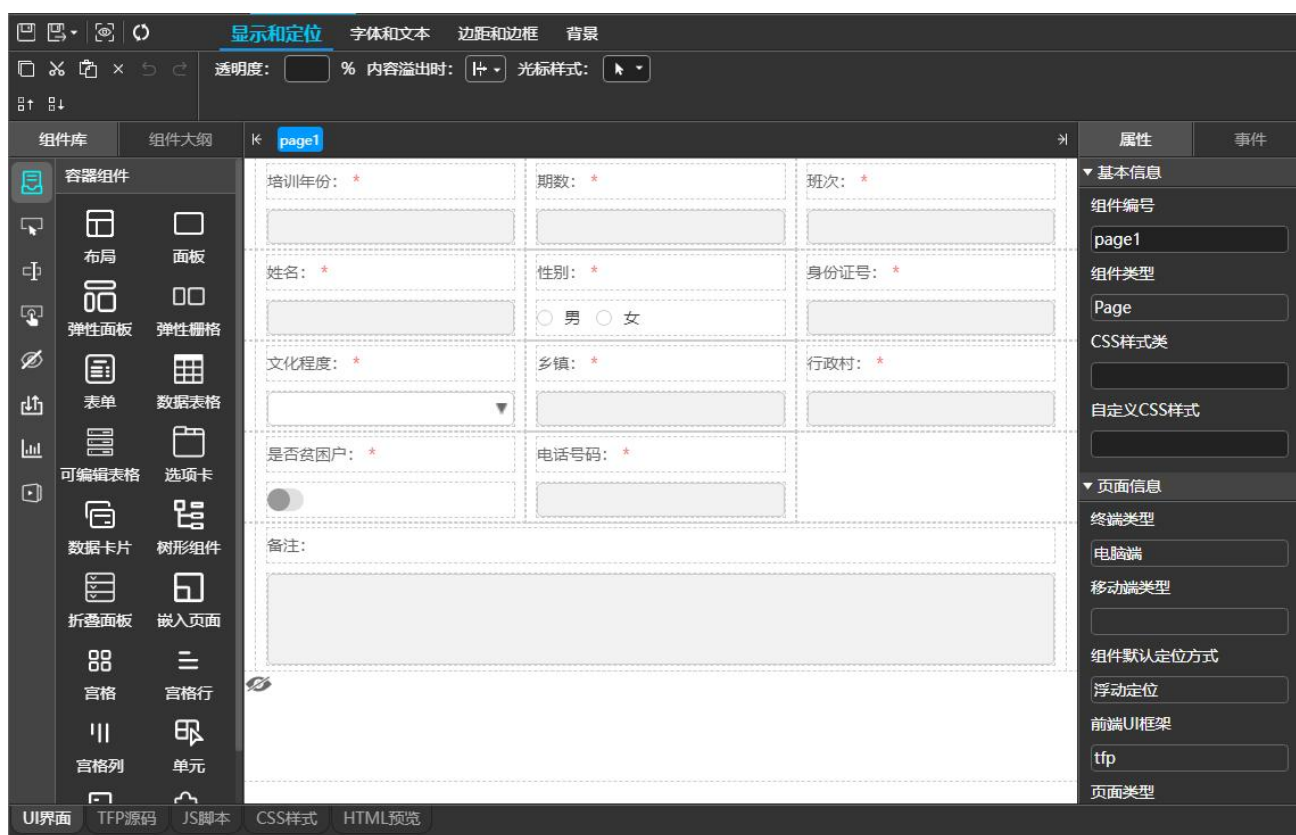
在“资源名称”输入框内输入：update，然后在下面的资源类型列表里点击“数据修改 TFP”按钮，然后点击“下一步”按钮，打开如下图所示的“创建数据修改 TFP”向导对话框：



在该对话框内，按以下步骤操作：

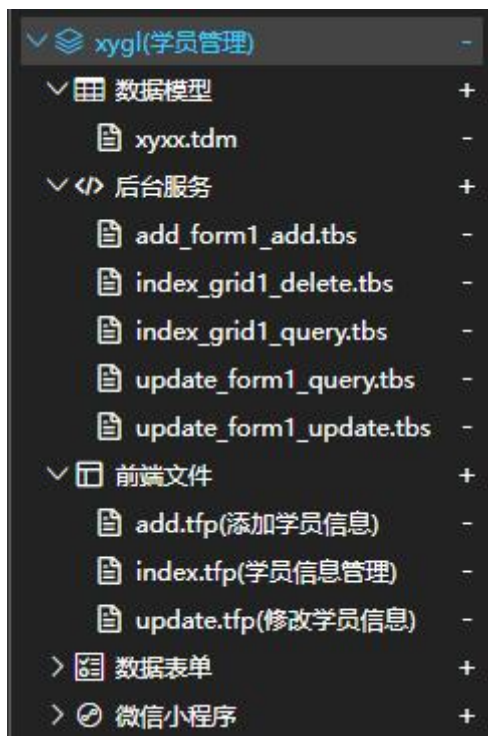
- (1) 先点击“数据模型”输入框右侧的“选择...”按钮，在打开的对话框里选择前面创建的“xyxx.tdm”这个数据模型，点确定按钮。
- (2) 在“页面标题”输入框内输入“修改学员信息”，其他选项默认即可，不用设置。
- (3) 点击字段列表里标题栏最左侧的复选框，全选所有字段，然后参考创建添加页面时的设置，设置各个字段的输入项类型、占用列数、是否必填和可选值。

然后点击“确定”按钮，即可完成学员信息修改页面的创建，TaskBuilder 会根据我们前面选择的字段及相关设置，自动创建对应的 tfp 页面以及输入项，如下图所示：



至此，我们就完成了学员信息管理应用的增删改查功能。

在 TaskBuilder 的项目资源管理器内，展开 xygl(学员管理) 这个项目的所有子节点，会看到下图所示的目录结构：

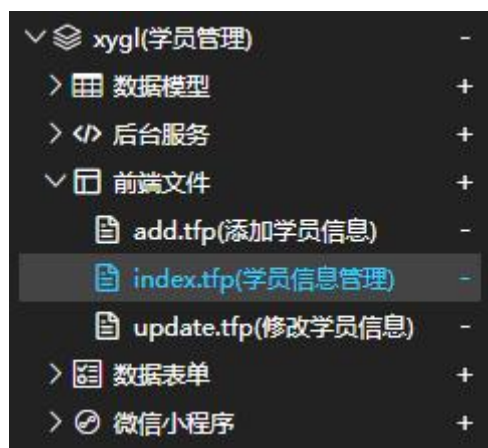


如果您看到的内容与上图不一致，可以在 xygl 项目节点上点鼠标右键，然后点刷新，然后再展开各个子节点。

从上图可以看到，通过上面的操作，向导为我们创建了数据模型、后台服务和前端文件这三种类型的文件，这三种文件是运行 TaskBuilder 开发的应用所必需的文件，后续会对这三种文件的功能进行详细介绍。

## 7、预览和测试

通过前面的操作，我们已经创建好了一个简单的学员信息管理应用，现在，我们点击项目资源管理器前端文件里的 index.tfp，如下图：





在打开的前端页面设计器里，点击顶部工具栏上的“预览”按钮：



会打开列表页面的预览窗口，如下图所示：



**说明：**预览时，服务器需要编译该页面的相关内容，会稍微有些慢，请耐心等待一下；正式运行时，服务器会缓存页面最后一次改动后编译的代码，不需要每次都编译，打开速度会很快。

在该窗口内，左侧为页面预览区域，模拟显示页面的实际运行界面，右侧是浏览器的调试窗口，可以查看控制台输出、页面元素和样式等，方便开发调试。

在该页面里，点击“添加”按钮，会打开添加学员信息的抽屉页，如下图所示：

刷新

添加

文化程度:

培训年份

期数

班次

添加

确定

取消

培训年份: \*

2024

期数: \*

第1期

班次: \*

2班

姓名: \*

李丽

性别: \*

☐ 男 ☒ 女

身份证号: \*

123456789012345678

文化程度: \*

高中

乡镇: \*

漳江镇

行政村: \*

胜利村

是否贫困户: \*

☐

电话号码: \*

13245678999

备注:

录入学员信息后，点击右上角的确定按钮，完成学员信息的添加，系统会自动跳转到列表页，就可以看到刚才添加的学员信息，如下图所示：

刷新

本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

添加

培训年份:

期数:

姓名:

身份证号:

文化程度:

查询

| 培训年份 | 期数  | 班次 | 姓名 | 性别 | 身份证号               | 文化程度 | 乡镇  | 行政村 | 是否贫困户 | 电话号码        | 修改 | 删除 |
|------|-----|----|----|----|--------------------|------|-----|-----|-------|-------------|----|----|
| 2024 | 第1期 | 2班 | 李丽 | 女  | 123456789012345678 | 高中   | 漳江镇 | 胜利村 | 是     | 13245678999 | 修改 | 删除 |

<<

<

1

>

>>

到第 1 页

共 1 页, 1 条记录

20

下面，我们再点击该学员信息所在行右侧的“修改”按钮，可以打开学员信息修改页面，如下图所示：

刷新

添加

文化程度:

查询

培训年份

期数

班次

姓名

性别

修改

确定

取消

培训年份: \*

2024

期数: \*

第1期

班次: \*

2班

姓名: \*

李丽

性别: \*

☐ 男 ☒ 女

身份证号: \*

123456789012345678

文化程度: \*

高中

乡镇: \*

漳江镇

行政村: \*

胜利村

是否贫困户: \*

☐

电话号码: \*

13245678999

备注:



我们修改一下期数、班次和文化程度这三项信息，然后点击确定按钮，回到列表页就可以看到修改后的信息已生效，如下图所示：

刷新 本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

添加 培训年份: 期数: 姓名: 身份证号: 文化程度: 查询

| 培训年份 | 期数  | 班次 | 姓名 | 性别 | 身份证号               | 文化程度 | 乡镇  | 行政村 | 是否贫困户 | 电话号码        | 修改 | 删除 |
|------|-----|----|----|----|--------------------|------|-----|-----|-------|-------------|----|----|
| 2024 | 第3期 | 3班 | 李丽 | 女  | 123456789012345678 | 大专   | 漳江镇 | 胜利村 | 否     | 13245678999 | 修改 | 删除 |

<< < 1 > >> 到第 1 页 共 1 页, 1 条记录 20

如果要删除学员信息，就在列表页面里，点击该学员信息所在行右侧的“删除”按钮，即可删除该学员。如果学员信息比较多，可以使用列表页面顶部的查询条件，按培训年份、期数、姓名和文化程度等条件查询。

## 8、发布应用

应用开发完，并经过测试没有问题后，就可以发布了。

发布应用需要登录任擎后台管理系统，操作步骤如下：

打开浏览器（推荐使用谷歌 Chrome 浏览器），在地址栏内输入任擎后台管理系统的网址，然后按回车，即可打开任擎后台登录页面，在任擎服务器所在电脑上访问后台管理系统的地址如下：

<http://localhost:8001/>

如果任擎服务器不是在当前电脑，则需要将上面网址中的“localhost”换成任擎服务器所在电脑的 ip 地址或域名。如果任擎服务器的网络端口不是 8001，则上面网址中的“8001”要改成任擎服务器实际使用的端口。

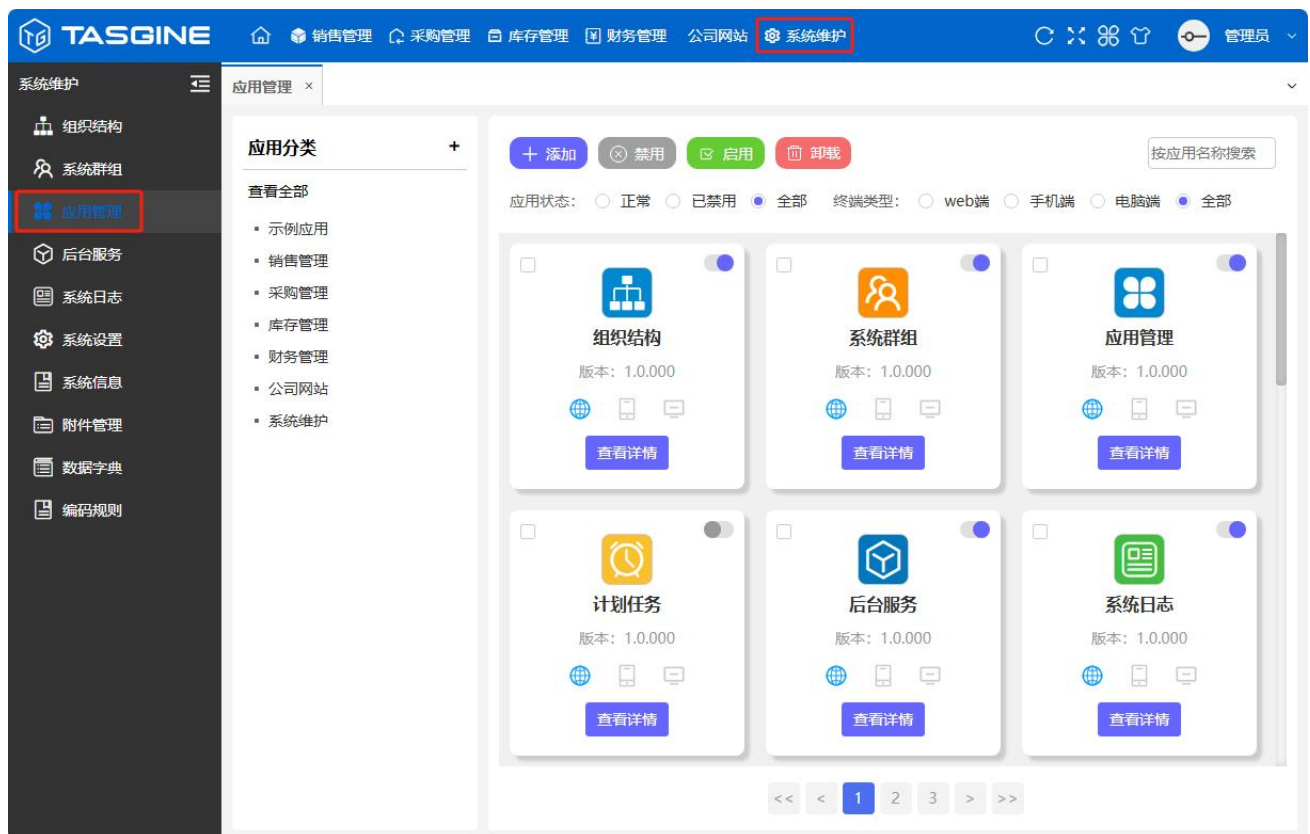
任擎后台管理系统的登录界面如下图所示：



输入用户名：admin，密码：taskmsg（这是默认密码，如果改过，则需要输入修改后的密码），然后点登录按钮，即可进入任擎后台管理界面，如下图所示：



点击系统顶部导航栏里的“系统维护”，然后再点击左侧菜单里的“应用管理”，即可打开系统应用管理模块界面，如下图所示：



再点击该页面顶部的“添加”按钮，打开添加应用的页面，如下图所示：

### 添加应用

应用图标: \*

选择

应用代码: \*

应用名称: \*

所属类别: \*

当前版本: \*

1.0.0.1

支持终端: \*

☐ web ☐ 任讯手机客户端 ☐ 任讯电脑客户端

描述:

提交

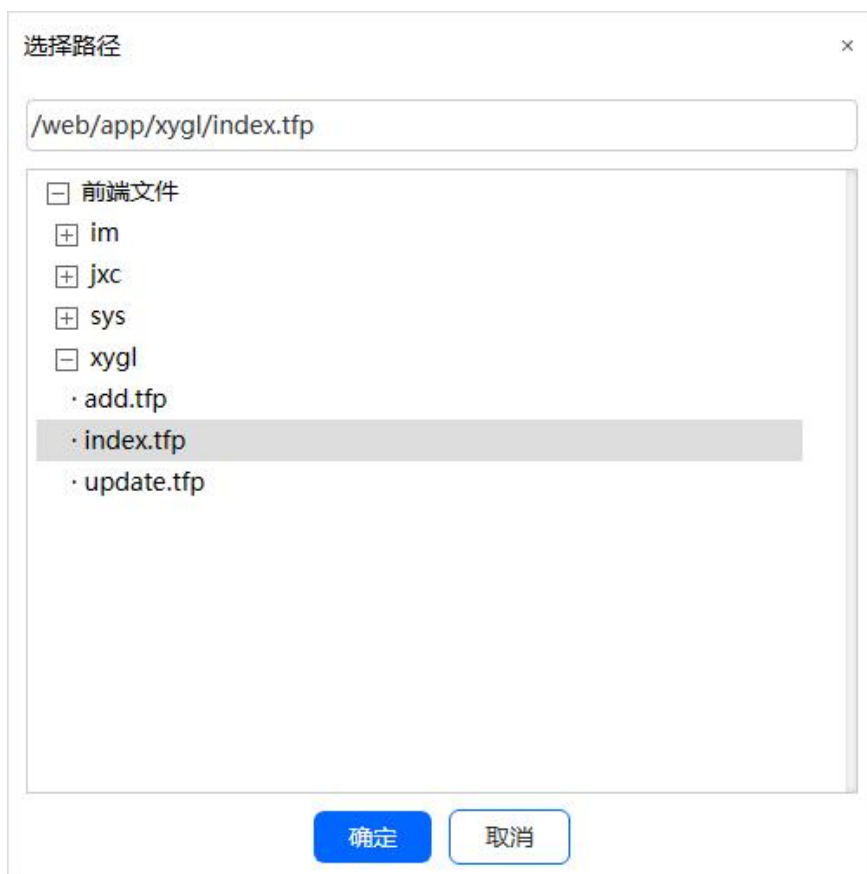
首先点击该页面“应用图标”文字标签右侧的“选择”按钮，打开图标选择窗口，如下图所示：



在该窗口中，先在左侧图标列表中选择适合当前应用的图标，然后在右侧设置背景颜色，然后再点击“图标颜色”，选中白色，然后点底部的“保存”按钮，即可完成应用图标的设置。

然后再输入以下信息：

- **应用代码：**xygl
- **应用名称：**学员管理
- **所属类别：**暂时先选“系统维护”
- **当前版本：**默认为“1.0.0.1”，可以不用改
- **支持终端：**选中“web”，表示该应用的页面是在电脑端 Web 浏览器内打开
- **web 端首页 URL 路径：**点击该输入框右边的“选择”按钮，可以打开前端页面路径选择窗口，找到 xygl 项目内的 index.tfp 页面，点确定，即可完成应用首页路径的设置，操作界面如下图所示：



都设置好之后的界面如下图所示：

添加应用

应用图标：\* 选择



应用代码：\* xygl

应用名称：\* 学员管理

所属类别：\* 系统维护

当前版本：\* 1.0.0.1

支持终端：\* ☒ web ☐ 任讯手机客户端 ☐ 任讯电脑客户端

web端首页URL路径：\* /app/xygl/index.tfp 选择

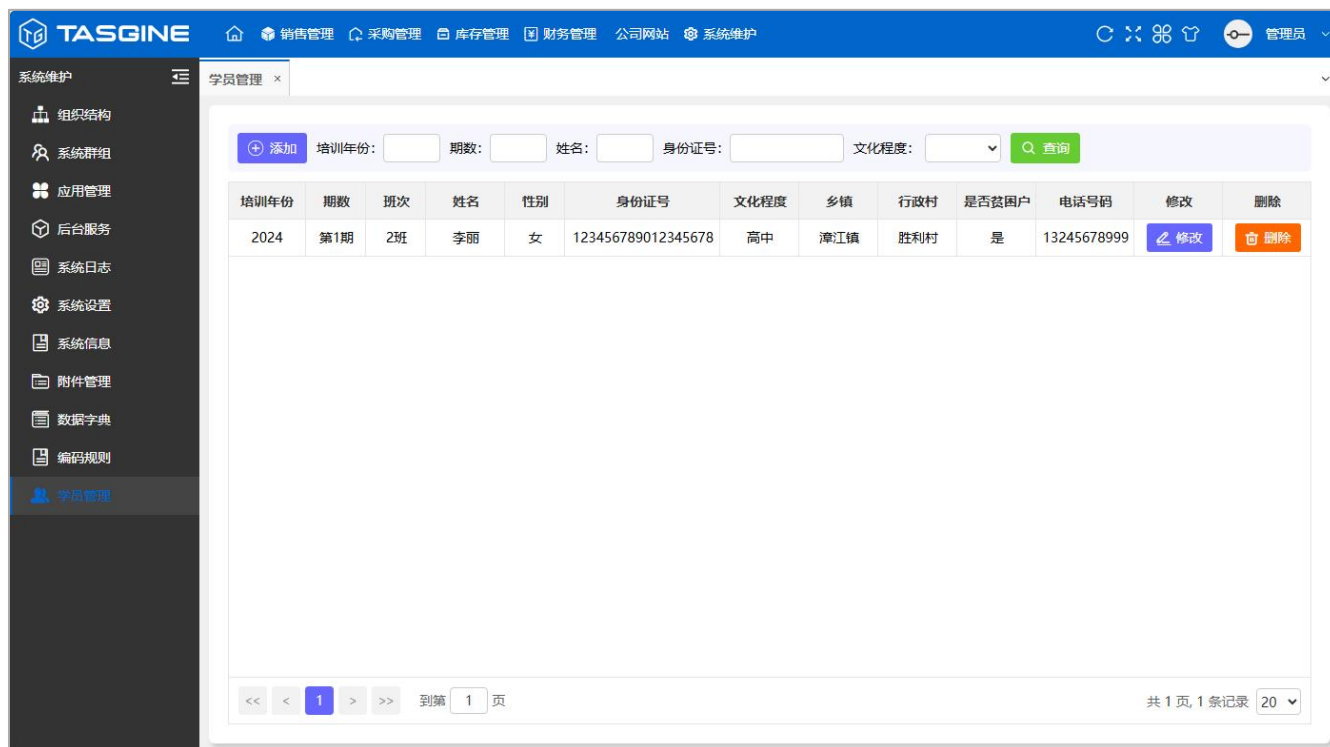
web端详细URL路径： 选择

描述：

提交

点击页面底部的提交按钮，即可完成“学员管理”这个应用的发布。

点击 Web 浏览器的刷新按钮，再点击后台管理系统顶部的系统维护，就可在系统左侧的应用列表里，看到刚发布的“学员管理”应用。点击该应用，即可打开学员信息列表页面，如下图所示：



至此，我们就完成了一个简单的“学员信息管理”应用的创建，大家觉得是不是很容易，但不要骄傲，这只是一个非常简单的应用，在我们实际工作中需要开发的应用要远比这个示例应用复杂得多，而且，上面只是介绍了如何按照向导一步一步完成应用的开发，并没有详细介绍这些功能具体的实现原理，还需要大家跟着本教程继续学习。

## 六、TaskBuilder 低代码开发基本原理解析

在继续介绍 TaskBuilder 的各项功能前，我们先讲解一下使用 TaskBuilder 进行低代码开发的一些基本原理。

### 1、信息系统常见的系统架构

#### 1.1 单文件架构



现在很多企业内部虽然已经建设了一些信息系统，但还是有不少业务没有用专门的信息系统管理起来，普遍都是采用 Excel 表格来实现这些业务数据的填报和查询统计。Excel 就是属单文件架构，这种架构是指整个系统就是一个文件，打开文件就可以直接操作，优点是简单易用，电脑上安装相应的软件就能使用，不需要专门的服务器，适合个人办公使用，如果有多个人需要查看或操作同一个表格文件时，就很不方便，需要把文件传来传去，每个人改完，还得跟其他人改的内容手动合并，不仅效率很低，而且很容易出错。

## 1.2 两层架构

二十年前，随着 PC 电脑的普及，很多企事业单位内部开始建设信息系统，那时候的信息系统基本都是采用 C/S 模式的两层架构，C 是指客户端软件（Client），S 是指服务器端软件（Server），服务器一般都是指关系型数据库，就是客户端软件通过网络直接连接数据库进行数据的增删改查操作。这种架构相对上面的单文件架构来说，可以实现功能更加强大和更加完善的信息系统，支持多人同时操作同一个业务的数据，而且，支持数据格式和业务逻辑校验，防止无效数据录入；支持操作权限管控，防止非法操作。

但随着信息化建设的不断发展，使用信息系统的人越来越多，这种模式的弊端也逐步显现出来，就是在用户量比较大时，这种客户端软件直接连数据库的模式会造成整个系统卡顿，甚至宕机。因为这种模式，每个客户端都会与数据库至少建立一个连接，且一直保持，当用户比较多时，就会建立大量的数据库连接，造成数据库性能急剧下降，最终影响整个系统的访问，所以，这种模式现在已经很少见了。

## 1.3 多层架构

现在的信息系统基本都是采取多层架构，最典型的就是三层架构，即：客户端软件+应用



服务器+数据库，我们常说的 B/S 架构就属于这种，B 是指 Web 浏览器（Browser），S 是指应用服务器（包含 Web 服务器），实际后面还有关系型数据库，采用 HTTP 协议通信的手机 APP、微信小程序等，也可以看做是 B/S 架构里的客户端，因为他们可以和 Web 页面共用服务器。

采用这种架构，客户端和数据库之间增加了一个应用服务器，客户端不能直接连接数据库，如果客户端需要对某项业务的数据进行增删改查操作，需要先连接到应用服务器，并发送具体的操作请求和参数，应用服务器进行相关校验后，再连接数据库执行实际的增删改查操作，并把结果返回给客户端，然后就关闭与数据库和客户端的连接，也就是说，应用服务器与数据库和客户端之间都不会一直连接，而是用完就关闭，这种方式可以大幅提升整个系统的性能。而且，这种架构把业务系统分成了前端界面和后台服务两个部分，前端界面负责填报和展示数据，后台服务负责执行必要的数据库校验和业务逻辑处理，并连接数据库执行实际的数据增删改查操作，前端代码由 Web 浏览器或手机 APP 等客户端软件执行，后台服务由应用服务器执行。这种将前后端解耦分别开发和执行的模式，一是可以让软件系统的各个部分分工更加清晰明确，各自可以专注去解决各自层面的问题，二是可以实现资源复用，例如同一个后台服务，可以供 Web 页面、手机 APP 或微信小程序调用，而不需要针对每种终端都重复开发。

对于大多数中小型信息系统来说，这种三层架构基本就能满足要求了，如果还想提升性能，可以增加缓存服务器，将需要频繁访问的热数据保存到缓存中，需要查询时，可以直接从缓存获取，从而减轻关系型数据库的压力，提升系统的承载能力。

对于大型信息系统来说，由于要承受大量并发访问，则上面这种简单的三层架构也无法满足要求，于是又衍生出了集群架构和微服务架构等多层架构，在这些架构下，服务器端不只是简单的一个服务器为客户端提供服务，可能有好多台服务器组成集群，然后在前面加上负载均衡服务器进行统一的业务受理和任务调度，从而可以承载更多的并发访问。采用这种架构，每台应用服务器上的代码都是一样的，当客户端请求某个后台服务时，由负载均衡服务器按照随

机、轮询等策略，将请求分派给其中一台处理，然后将处理结果返回给客户端。

如果采用微服务架构，还需要在应用服务器集群前面增加微服务网关，在网关中配置每个后台服务都由哪些服务器提供服务，从而可以实现针对不同的后台服务，可以分配不同的服务器或服务器集群进行处理，访问量小的，可以多个后台服务放在一台服务器上，访问量大的，可以单独放在一台服务器或者多台服务器上。在数据库层面，关系型数据库和缓存数据库也都可以由多台服务器组成集群。另外，根据需求，还可以部署单独的消息服务器、身份认证和鉴权服务器等。这些服务器端软件可以部署在物理主机或虚拟主机上，也可以部署在容器内，总之，部署方式可以非常灵活，可以按需扩容，是目前很多大型互联网平台普遍采用的架构。

## 1.4 TaskBuilder 支持的系统架构

用 TaskBuilder 开发的应用具体采用的什么架构取决于使用的什么版本的任擎，共享版的任擎采取的是最简单的三层架构，即在服务器端，由一个单独的任擎进程统一为客户端提供所有服务，包括静态文件访问、动态服务执行、数据库操作、身份验证、数据缓存等等，可以满足中小型信息系统的需求。

如果系统用户较多，并发访问量较大，则可以使用任擎企业版或集群版。

任擎企业版支持在一台服务器上运行多个任擎进程，一个主进程，多个从进程，由主进程统一受理客户端请求，然后再分发给从进程进行处理，处理完再将结果返回给主进程，主进程再返回给客户端，从而实现负载均衡，满足大量用户的并发访问。并且，主进程负责实时监控各个从进程的运行状态，如果发现某个从进程出现异常，可以主动重启该从进程，保证整个系统能长期稳定运行。

任擎集群版支持基于微服务架构的分布式部署，可以将整个系统的各个后台服务分别部署在多个主机或容器中，可以根据各个后台服务的负载灵活分配服务器资源或弹性扩容，可以支

持海量用户的并发访问，适合用来运行大型信息系统或互联网平台。

不管使用什么版本的引擎，核心功能都是一样的，只是架构不一样，将引擎从一个版本切换为另外一个版本时，用 TaskBuilder 开发的应用不需要进行任何修改，可以平滑切换。

## 2、前端文件和后台服务的概念

多层架构的应用系统里文件会分为前端文件和后台服务，这两类文件的具体含义如下：

### 2.1 前端文件

在 Web 浏览器、手机 APP 或微信小程序等客户端软件中运行的代码(包括 HTML、JavaScript 和 CSS 等)或图片、音视频等资源文件统称为前端文件，用来填报和展示信息。如果客户端是 Web 浏览器或手机 APP 里嵌入的浏览器，则前端文件是保存在服务器上的，当浏览器访问指定地址的网页时，会向 Web 服务器请求该页面的内容，Web 服务器从服务器硬盘上加载这些文件的内容，然后直接返回给浏览器，浏览器再加载执行或显示，Web 服务器并不会执行这些文件里的代码，所以这些文件也称为：**静态文件**。如果客户端是手机 APP 原生代码或微信小程序，则前端文件是直接编译打包在客户端软件中的，不需要从 Web 服务器加载，只需要向服务器请求动态数据。

### 2.2 后台服务

在服务器上由 Web 服务器软件加载和运行的代码称为后台服务，客户端请求后台服务时，Web 服务器不是直接把这些文件里的内容返回给客户端，而是在服务器上加载并运行，然后将执行结果返回给客户端，所以这些文件也称为：**动态文件**。后台服务主要用来查询、添加、修改和删除关系型数据库中的数据，也可以实现导入导出、文件上传下载、发送邮件或短信、

与第三方系统对接等更多操作。

为了区分仅能提供静态文件浏览服务的简单 Web 服务器，我们一般将能执行动态后台服务代码的 Web 服务器称为：**应用服务器**，或者 **中间件服务器**，我们任擎就是属于这种既可以提供静态文件浏览，又能执行动态后台服务的应用服务器。

任擎目前支持的后台代码语言为 JavaScript，如果后台要采用其他语言开发，可以通过微服务网关或反向代理等技术实现。

### 3、TaskBuilder 低代码开发必须的三种文件

用 TaskBuilder 开发的应用系统属于多层架构，也有前端文件和后台服务，但是前后端除了能直接执行的代码文件以外，还多了 **设计时** 的文件，因为 HTML 代码和 JavaScript 代码的结构太复杂，很难精确地还原为可以可视化设计的界面，所以 TaskBuilder 将可视化设计的前后端文件都保存为了 JSON 格式的 **设计时** 文件，以便后续还能用 TaskBuilder 打开并用相应的图形化设计器查看或继续进行修改编辑。

另外，为了方便在 TaskBuilder 中进行可视化开发时，能快速获取数据库中表或视图的结构信息，还需要创建数据模型文件。

总之，数据模型文件、后台服务文件和前端页面文件是使用 TaskBuilder 进行低代码开发必须的三种文件，下面分别对这三种文件进行详细介绍。

#### 3.1 数据模型文件

TaskBuilder 主要用来开发基于关系型数据库的信息系统，在开发的过程中，经常需要与数据库打交道，为了方便开发者对数据库中的表和视图进行统一的维护和管理，TaskBuilder 提供了专门的数据模型设计和管理的功能，可以直接在 TaskBuilder 里设计表和视图对应的数据模型，然后同步到数据库，也可以将数据库中已有的表或视图导出为数据模型文件。

通过数据模型，一是可以在开发的过程中快速获取表或视图的定义信息，二是可以在数据模型文件中配置一些常用的校验规则和关联关系，以便在对这些表或视图进行操作时能自动进

行数据校验和关联更新。

### 3.1.1 数据模型文件的格式

TaskBuilder 中的数据模型文件的扩展名为.tdm (Taskmsg Data Model)，用来定义应用的数据结构，对应数据库里的表或视图，该文件的实际内容为 JSON 格式，里面定义的信息包括数据库类型、数据源 id、数据对象类型（表或视图）和名称，以及字段信息列表等，字段信息包括字段的名称、数据类型、长度、是否允许为空、默认值是多少等等。在数据模型设计器内，点击底部的“TDM 代码”选项卡，即可查看数据模型的 JSON 代码，如下图所示：

```
1 {
2   "table": "sys_app_attach",
3   "type": "table",
4   "fields": {
5     "id": {
6       "dataType": "INT",
7       "type": "int",
8       "primaryKey": true,
9       "autoIncrement": true,
10      "notNull": true,
11      "uniqueIndex": false,
12      "length": 11,
13      "default": "",
14      "comment": "",
15      "version": 1,
16      "status": 0,
```

### 3.1.2 数据库表的模型设计

如果数据模型对应的是数据库中的表，则打开后的数据模型设计器界面如下图所示：

| <input type="checkbox"/> | 字段名       | 字段类型 | 长度  | 精度 | 主键                                  | 非空                                  | 自增                                  | 唯一                       | 默认值 | 备注   |                                                                                                    |
|--------------------------|-----------|------|-----|----|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-----|------|----------------------------------------------------------------------------------------------------|
| <input type="checkbox"/> | id        | 整数   | 11  |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |     | 客户编号 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | code      | 字符串  | 10  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 客户编码 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | name      | 字符串  | 200 |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 客户名称 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | scale     | 字符串  | 4   |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 客户规模 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | trade     | 字符串  | 4   |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 所属行业 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | grade     | 字符串  | 4   |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 客户等级 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | source    | 字符串  | 4   |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 客户来源 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | telephone | 字符串  | 45  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 联系电话 | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |
| <input type="checkbox"/> | contacts  | 字符串  | 20  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 联系人  | <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↻"/> |

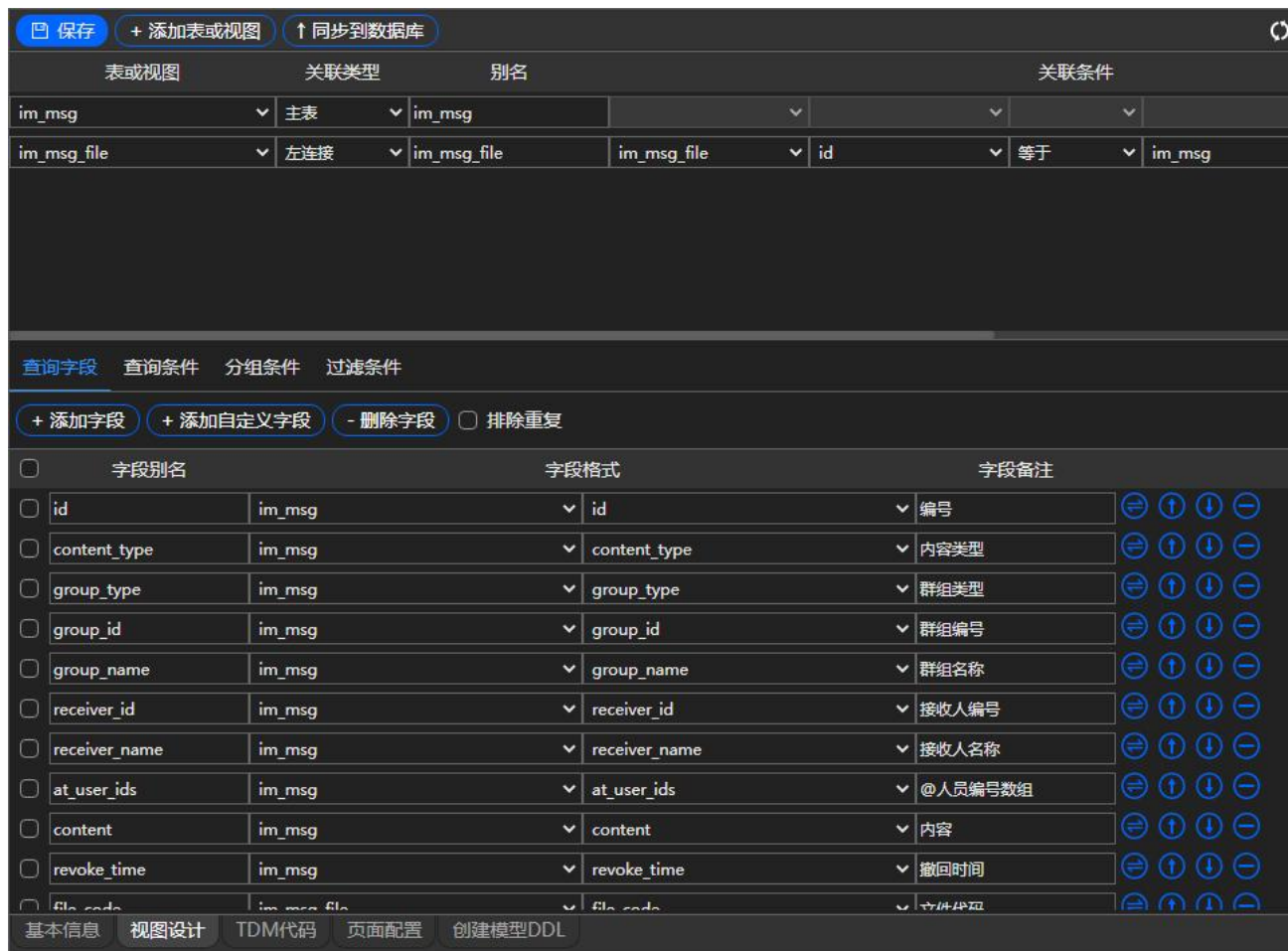
基本信息 字段列表 TDM代码 创建模型DDL 变更模型DDL

在此可以设置数据模型具体包括有哪些字段，以及各个字段的数据类型、长度、是否是主

键、是否非空、是否自动增长、是否唯一、默认值、备注等信息。

### 3.1.3 数据库视图的模型设计

如果数据模型对应的是数据库中的视图，则打开后的数据模型设计器界面如下图所示：



在此可以设置数据模型具体由哪些表关联构成，哪个是主表，各个表之间的关联条件，视图有哪些字段，各个字段是属于哪个表的，另外，还可以设置查询条件、分组条件和过滤条件。

### 3.1.4 数据模型的双向同步

如果在数据库里修改了表或视图，可以点工具栏上的“从数据库同步”按钮将改动同步到数据模型文件中，如果在数据模型设计器中修改了数据模型，可以点工具栏上的“同步到数据库”按钮将改动同步到数据库。

### 3.1.5 查看数据模型基本信息

在数据模型设计器内，点击底部的“基本信息”选项卡，即可查看当前数据模型的基本信

息，如下图所示：

|         |                           |
|---------|---------------------------|
| 模型文件名：  | pm_financial_accounts.tdm |
| 数据源ID：  | default                   |
| 数据库类型：  | mysql                     |
| 数据对象类型： | 表                         |
| 同步状态：   | 已同步                       |
| 模型备注：   | 财务账户                      |

在此可以查看数据模型的文件名、数据源 ID、数据库类型、数据对象类型、同步状态和模型备注，其中，模型备注可以设置和修改，如果设置了备注，则在项目资源管理器的数据模型列表里，该模型文件名后面会显示备注信息。

### 3.1.6 查看创建模型的 DDL 语句

在数据模型设计器内，点击底部的“创建模型 DDL”选项卡，即可查看创建该数据模型的 DDL SQL 语句，如下图所示：

```
保存 另存为 复制SQL ↓从数据库同步 ↑同步到数据库
1 CREATE TABLE `pm_financial_accounts` (
2   `id` INT NOT NULL AUTO_INCREMENT COMMENT '编号',
3   `code` VARCHAR(20) NOT NULL COMMENT '编码',
4   `name` VARCHAR(45) NOT NULL COMMENT '名称',
5   `type` VARCHAR(10) NOT NULL COMMENT '类型',
6   `bank` VARCHAR(200) COMMENT '开户银行',
7   `account` VARCHAR(45) COMMENT '银行账号',
8   `remaining_sum` DECIMAL(16,2) COMMENT '余额',
9   `status` SMALLINT COMMENT '状态',
10  PRIMARY KEY(`id`)
11 ) ENGINE=INNODB CHARSET=UTF8MB4 COMMENT='财务账户';
```

点击工具栏上的“复制 SQL”按钮，然后粘贴到其他数据库管理工具内执行，即可在其连接的数据库内创建当前数据模型定义的表或视图。



### 3.1.7 使用数据模型的必要性

有同学可能会问，既然数据模型文件里保存的都是表或视图的信息，那进行以上操作时，为什么不直接从数据库里获取表或视图的信息呢？主要有以下几个原因：

- **性能**：数据模型文件的内容会被缓存到内存中，需要使用时可以直接从内存读取，比从关系型数据库查询要快得多；
- **扩展性**：数据模型文件里可以添加一些数据库里没有的信息，例如数据模型所属的数据源、多个表之间的关联关系、字段的扩展校验规则等；
- **兼容性**：数据模型文件是一个中性的文件，理论上可以适配任擎支持的任何数据库。

总之，数据模型是使用 TaskBuilder 进行低代码开发所需的最基础信息，也是必须的信息，用 TaskBuilder 开发应用的第一步就是先创建好数据模型，否则后续的工作都无法进行。

## 3.2 后台服务文件

本文所说的 **后台服务**(Service) 是指在服务器电脑上由任擎(tasgine)软件执行的代码，它可以根据客户端发送的请求数据，进行数据库增删改查、数据导入导出、文件上传下载、文件读写等操作，并向客户端返回处理结果。后台服务运行时是没有界面的，占用的是服务器电脑的相关资源。

后台服务文件一般不需要开发者手动创建，使用向导创建前端页面，或者在前端页面里为各个组件设置数据增删改查操作时，都会自动创建对应的后台服务文件。但如果后台服务的业务逻辑比较复杂，自动创建的无法满足要求，则可以手动创建并进行详细的设置。

### 3.2.1 后台服务文件的格式

任擎服务器目前支持编辑两种格式的后台服务：

- **js 格式**：扩展名为.js，内容为 JavaScript 代码，这种格式任擎可以直接加载运行，是后台服务最终运行时的代码格式，不需要编译，但只能使用代码编辑器编辑，不能可视化设计，本文档主要讲低代码开发，所以不会深入介绍这种纯代码格式的后台服务。

**tbs 格式**：扩展名为.tbs (Taskmsg Back-end Service)，实际内容格式为 JSON。这种格式支持在 TaskBuilder 内进行可视化设计，在客户端请求这种格式的后台服务时，任擎会将

该文件自动编译为 js 代码并加载运行。这种格式是一种中性的格式，理论上可以编译为任意编程语言，例如 Java、C#、Python 等。

js 格式的后台服务用代码编辑器打开后的界面如下图所示，在此可以直接编写后台服务的 js 代码：

```

1  var Service = require('Service');
2  var util = require('util');
3
4  /**
5   * 退出
6   */
7  var Quit = function() {
8      Service.call(this);
9  };
10
11  util.inherits(Quit, Service);
12
13  module.exports = Quit;
14
15  /**
16   * 处理服务请求
17   * @param {Object} req 服务请求对象
18   * @param {Object} res 服务响应对象
19   */
20  Quit.prototype.process = function(req, res) {
21      server.sessionManager.endSession(req.sessionId);
22      this.end(res);
23  };

```

在 tbs 格式的后台服务文件内，定义了具体有哪些操作及其详细的参数设置，如下图所示：

```

1  {
2      "name": "Pm_customer_Add",
3      "comment": "添加pm_customer",
4  >  "reqArgs": [ ...
111 ],
112 >  "resArgs": [ ...
117 ],
118  "methods": [
119      {
120          "name": "process",
121          "comment": "处理服务请求",
122          "args": [
123              {
124                  "name": "req",
125                  "type": "object",
126                  "comment": "服务请求对象"
127              },

```

**注意：**在 TaskBuilder 内，可以将 tbs 格式的后台服务另存为 js 格式的，但无法把 js 格式还原为 tbs 格式。

## 3.2.2 后台服务设计器简介

tbs 后台服务设计器由以下几个部分构成：

### 3.2.2.1 基本信息

后台服务的基本信息设置界面如下图所示：

| 参数名       | 参数备注 | 参数类型 | 不为空                                 | 默认值 | 忽略注入                     | 文本验证                 |
|-----------|------|------|-------------------------------------|-----|--------------------------|----------------------|
| code      | 客户编码 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| name      | 客户名称 | 字符串  | <input checked="" type="checkbox"/> |     | <input type="checkbox"/> | <input type="text"/> |
| scale     | 客户规模 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| trade     | 所属行业 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| grade     | 客户等级 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| source    | 客户来源 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| telephone | 联系电话 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| contacts  | 联系人  | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| site      | 公司网址 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |
| province  | 所在省份 | 字符串  | <input type="checkbox"/>            |     | <input type="checkbox"/> | <input type="text"/> |

在后台服务的基本信息里可以设置以下内容：

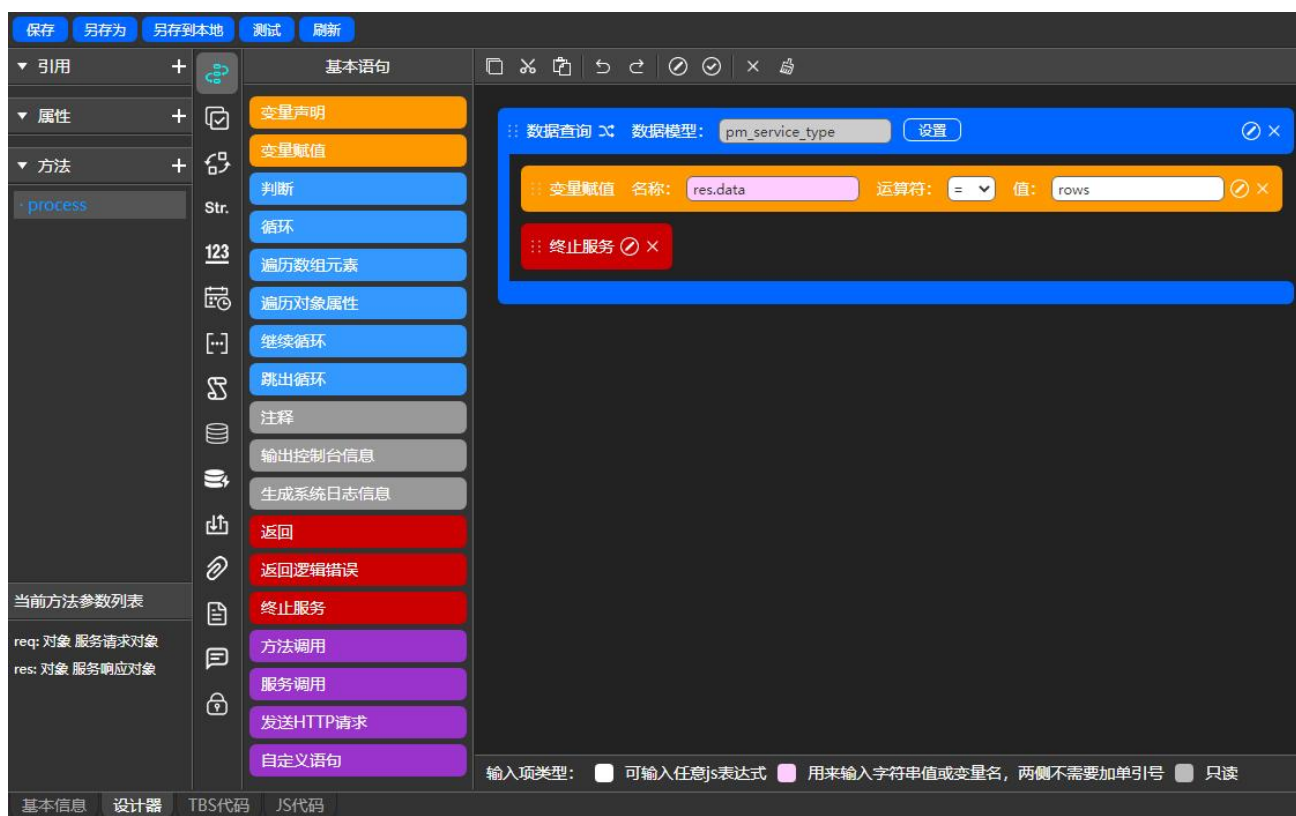
- **服务名称：**用来设置后台服务类的名称，一般不需要设置，创建后台服务时会自动生成，只能是字母、数字和下划线，且必须是字母开头，建议首字母大写。
- **服务说明：**用来设置后台服务的功能说明。
- **不校验是否登录：**后台服务默认都是需要登录才能访问的，如果某些后台服务不需要登录也可以访问，则需要在基本信息里，将“不校验是否登录”这个复选框勾上。
- **不检查 SQL 注入：**引擎默认会对客户端发送给后台服务的所有请求参数进行校验，判断参数中是否存在 SQL 注入的可疑字符，如果存在，则会自动拦截该请求，并向客户端返

回错误。如果您在开发某个应用时，前端页面提交的数据包含 insert、update、select 等 SQL 关键字，或者一些特殊字符，例如：\*、%、'、" 等，则可以在对应的后台服务的基本信息里，把“不检查 SQL 注入”这个选项选中。

- **请求参数：**在此可以设置客户端可能会提交给当前后台服务的所有请求参数，并设置这些参数的一些细节信息，包括：参数备注、参数类型、是否不能为空、默认值、是否忽略 SQL 注入、文本验证的正则表达式等。设置好这些信息后，当有客户端请求该服务时，引擎会自动对这些参数进行校验，包括数据类型是否正确、不能为空的参数是否传递了值、是否有 SQL 注入的非法内容、是否满足文本验证的正则表达式的要求等，如果校验未通过，会向客户端返回详细的错误信息。如果设置了参数备注，在错误信息里会使用备注来提示，否则用参数名提示。另外，如果设置了默认值，当前端传递的参数为空时，会使用默认值作为该参数的值。

### 3.2.2.2 业务逻辑设计器

tbs 后台服务的业务逻辑设计器如下图所示：

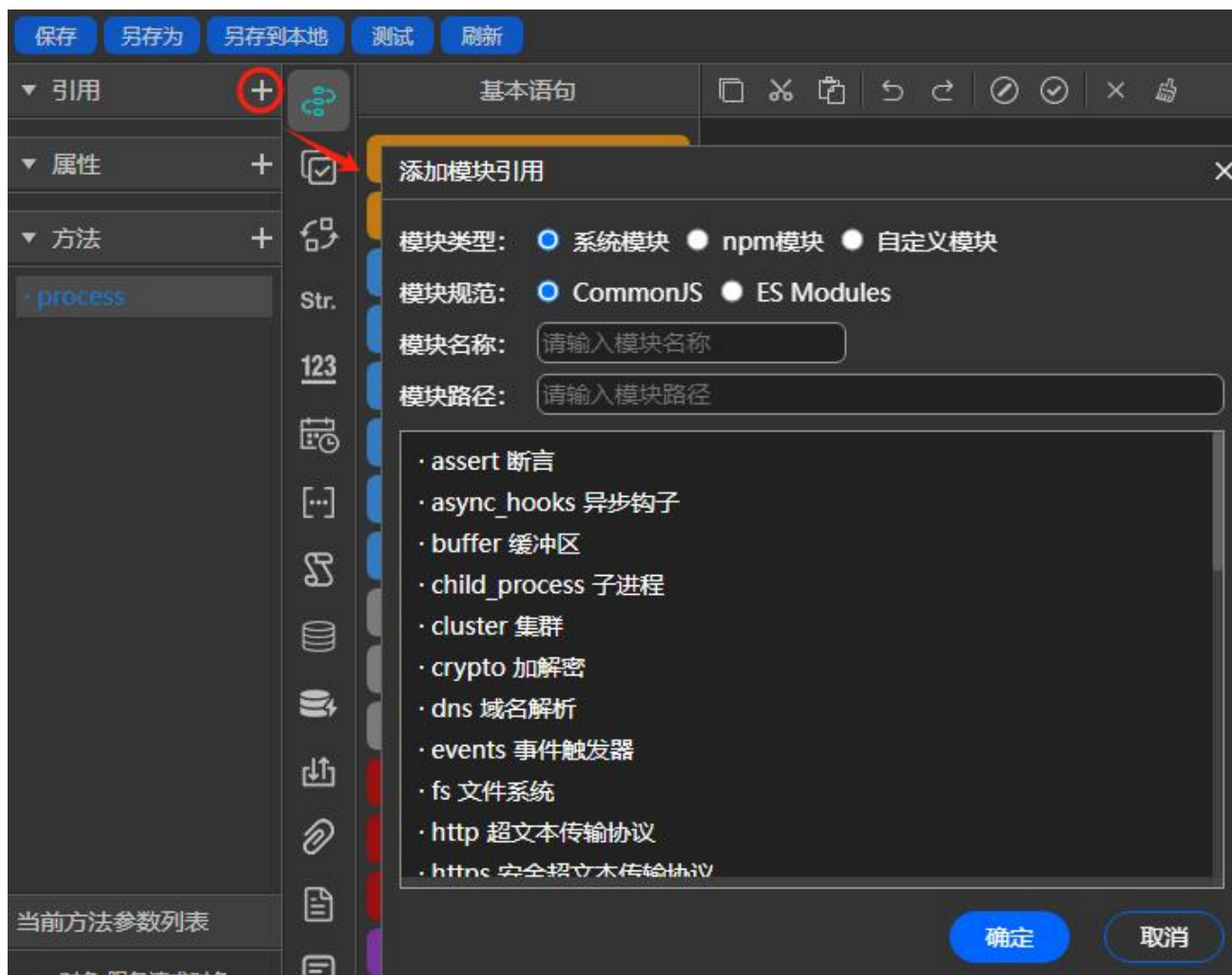


该设计器提供了丰富的功能，可以通过鼠标拖拽快速创建常用的后台操作，包括数据库增删改查、Excel 文件导入导出、附件上传下载、邮件和短信发送等。

tbs 后台服务的业务逻辑设计器由以下几个部分构成：

### 3.2.2.2.1 模块引用

如果在后台服务里需要用到第三方 js 模块（文件）的功能，则可以点击设计器左上角“引用”栏右侧的加号，然后弹出以下窗口：



在该对话框中，首先需要选择模块的类型，包括以下三种类型：

- **系统模块：**指 node.js 运行环境内置的模块，例如文件访问模块 fs、加解密模块 crypto、网络操作模块 net 等，具体有哪些内置模块以及如何使用请查看 node.js 的官方文档；
- **npm 包：**指通过 npm install 命令安装在 tasgine\node\_modules 目录里的 npm 包，npm 是 JavaScript 的包管理工具，让 JavaScript 开发人员可以轻松地使用其他开发人员共享的代码，开发者可以上网站 <https://www.npmjs.com/> 搜索自己所需的模块。如果要引用 npm 包，模块路径只需要设置为 npm 包的包名即可，引擎会自动去 tasgine\node\_modules 目录查找该包对应的模块文件，我们连接 MySQL 数据使用的



mysql 模块、导入导出 Excel 文件使用的 node-xlsx 模块就是属于这种 npm 模块。

- **自定义模块：**指部署在 tasgine\app\项目代码\service 目录里的各种后台业务服务，如果选择这种类型，上面的对话框中会显示当前引擎服务器上的项目列表，可以层层展开找到需要的服务，然后选中。

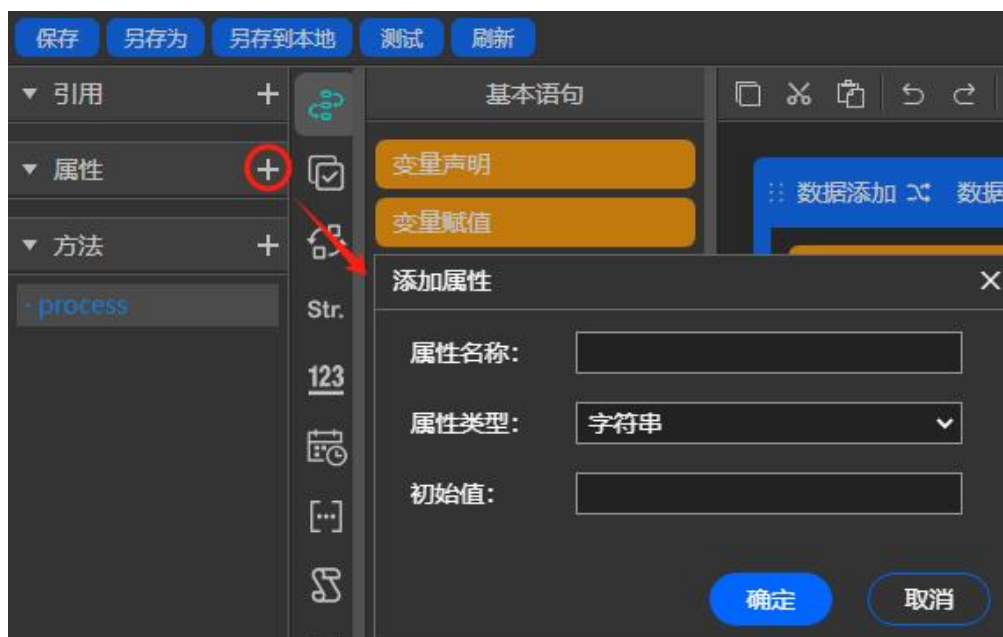
在后台服务中引入的 js 模块必须满足 CommonJS 或 ES6 这两种 js 模块规范中的一种，否则无法正确引入，CommonJS 是 Node.js 默认模块规范，Node.js 内置的系统模块、大部分 npm 模块和用 TaskBuilder 创建的后台服务都是采用的这种规范，有一些比较新的 npm 模块可能会采用 ES6 模块，具体哪个 npm 模块采用的什么规范请查看该模块的说明文档。

上图对话框中的模块名称是指在当前后台服务中为引用的模块分配的一个变量名，可以和 npm 包的名称一致，也可以改成其他名称，开发者可以根据自身需要进行设置。

模块路径是指模块的引用路径，如果是系统模块或 npm 模块，模块路径就是模块名称，不需要加其他目录名，如果是自定义模块，则需要在模块路径中提供相对于 tasgine\app 目录的完整路径，例如：/demo/service/test.js，否则调用该服务时，会出现无法找到服务的错误。

### 3.2.2.2.2 属性

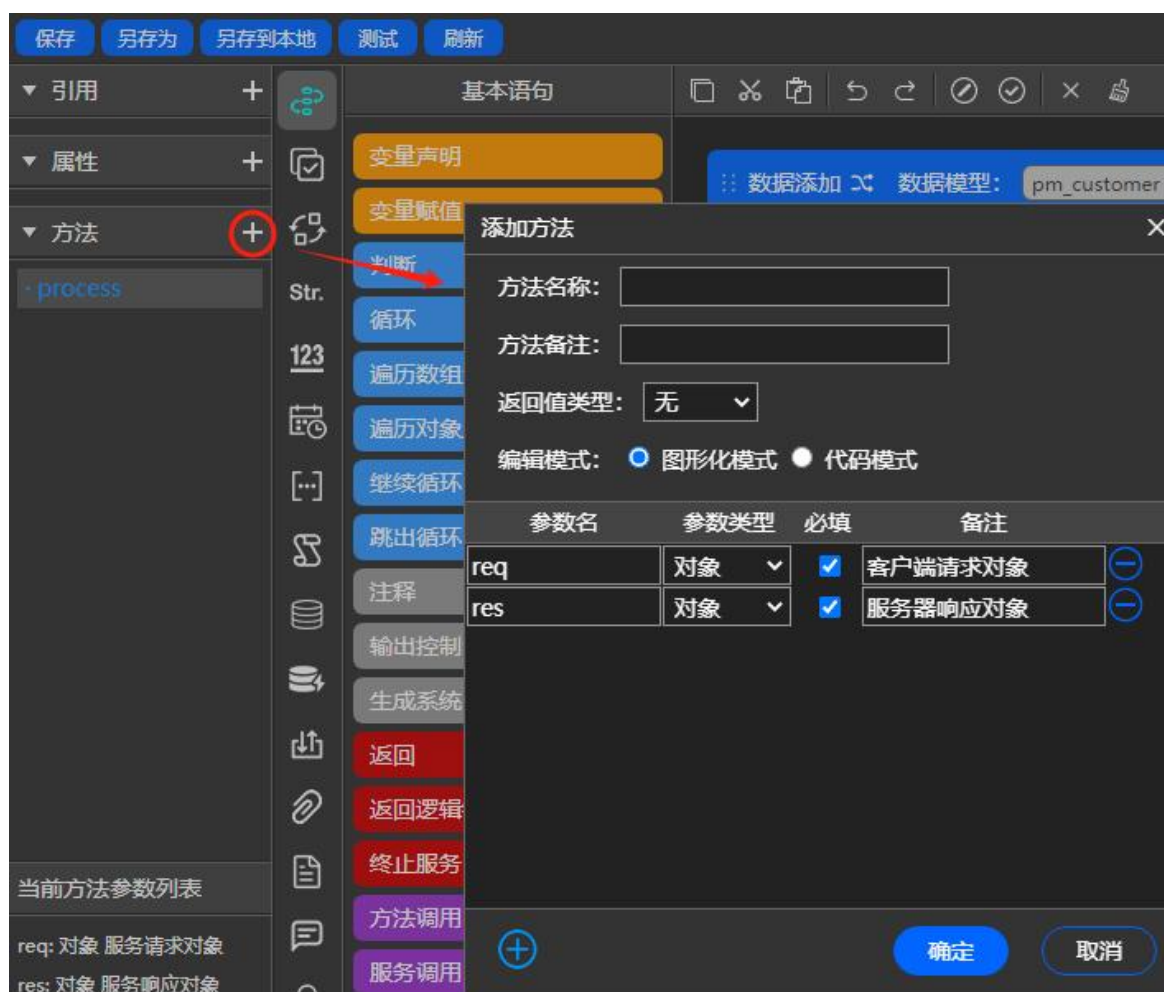
如果后台服务中定义了多个方法，而且需要在多个方法之间共享数据，则可以通过后台服务中的属性来实现，点击后台服务设计器左侧“属性”栏右侧的加号按钮，即可打开“添加属性”的对话框，如下图所示：



属性类似前端页面里的全局变量，只不过在后台服务的各个方法里访问时，需要在属性前面加上 **self.** 前缀，表示是当前后台服务对象的属性，而不是方法参数或方法内的局部变量。

### 3.2.2.2.3 方法

后台服务里的方法类似前端页面里的 js 函数，用来实现某个特定的功能。点击后台服务设计器左侧“方法”栏右侧的加号按钮，即可打开“添加方法”的对话框，如下图所示：

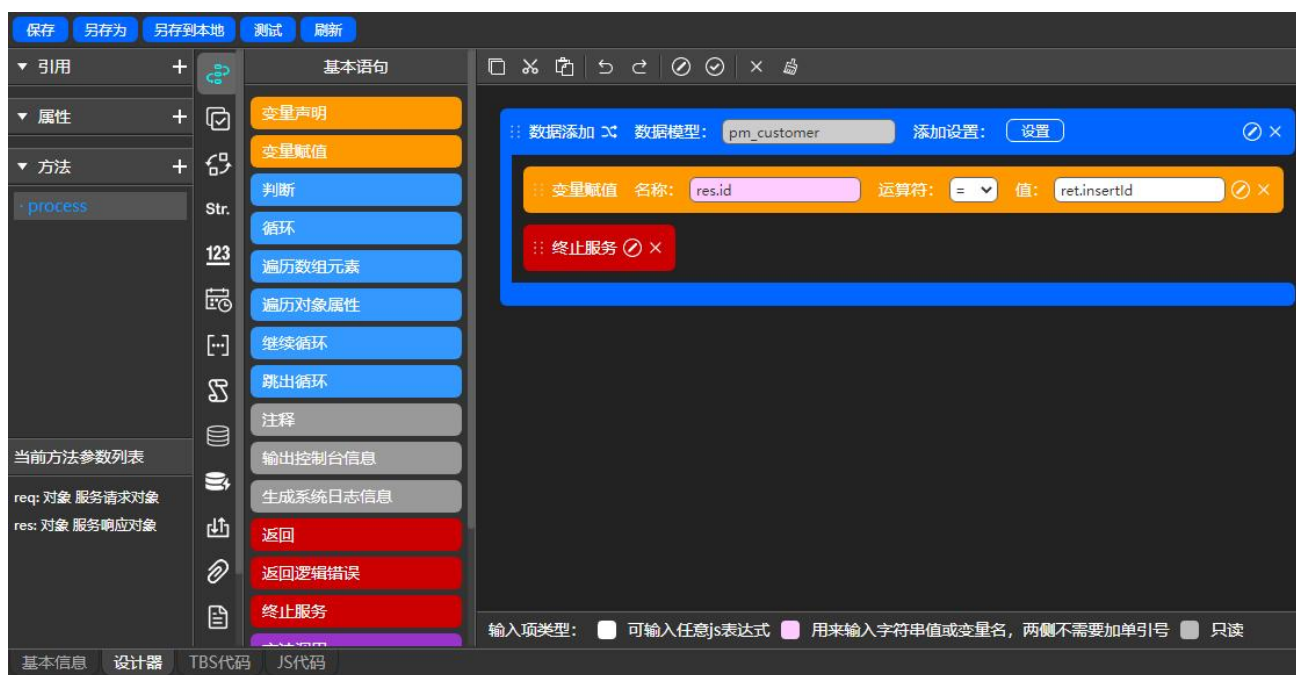


输入方法名称、备注，选择返回值类型和编辑模式，设置好调用该方法需要传入的参数，点确定即可完成方法的创建。

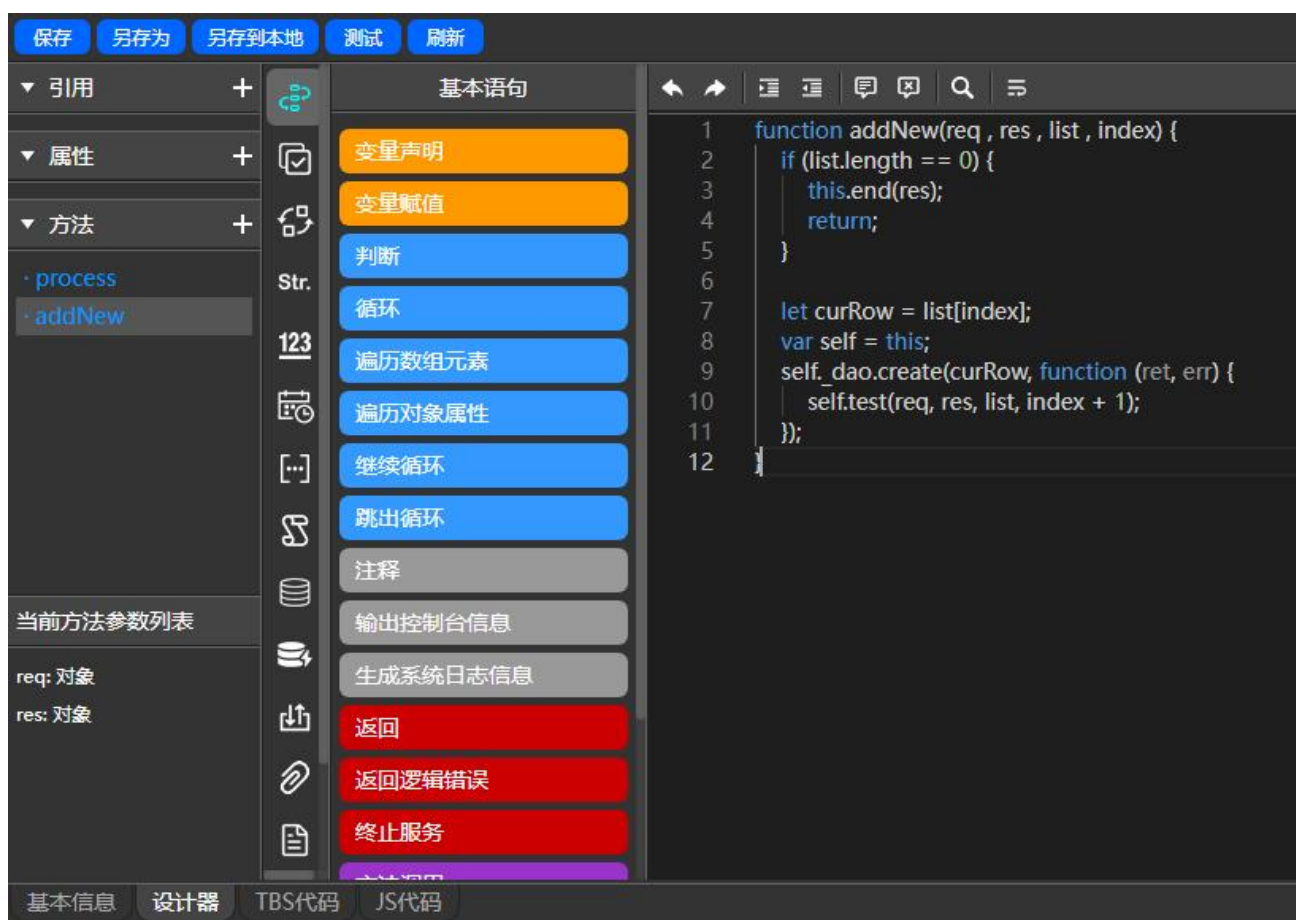
其中，方法的编辑模式支持图形化和代码两种模式，如果是图形化模式，会在设计器右侧显示一个可视化业务逻辑编排区，可以在此编排当前方法的业务逻辑，可以从左侧语句库中拖拽各种语句到右侧可视化业务逻辑编排区，并设置其详细参数，TaskBuilder 提供了一百多个常用的后台操作语句，包括基本语句、数据格式校验、数据格式转换、字符串处理、数值处理、日期时间处理、数组处理、系统数据、数据库操作、数据缓存操作、导入导出、附件操作、文



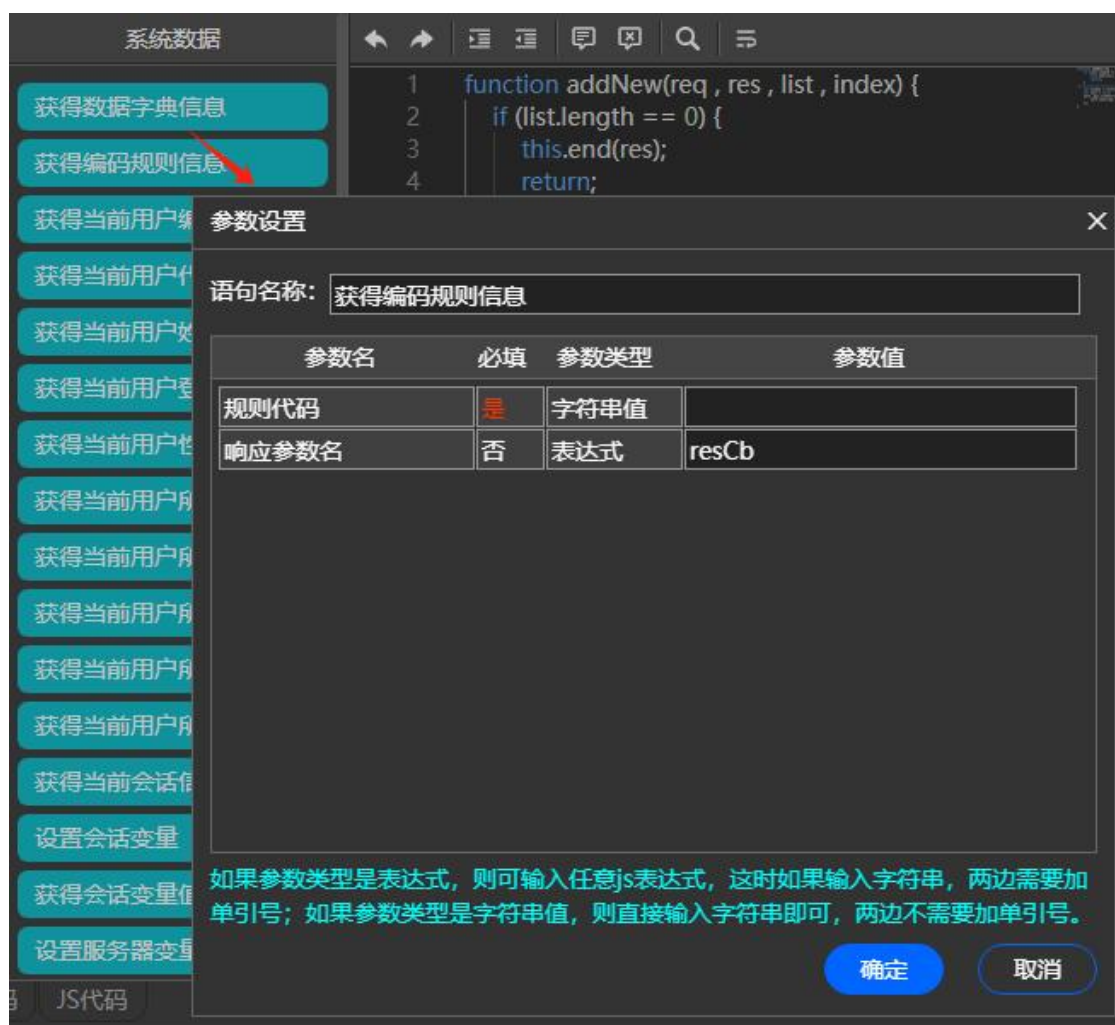
件操作、通讯和加密解密这几大类，如下图所示：



如果是代码模式，则只能在代码编辑器中编写当前方法的逻辑代码，如下图所示：



当然，也可以从左侧语句库中拖拽语句到代码编辑器，但需要先设置好语句的参数，点确定后，就会将该语句的相应代码插入到光标位置，插入完之后，就不能再图形化修改了。



用 TaskBuilder 创建的所有后台服务都默认有一个 `process(req, res)` 的入口方法，该方法为图形化模式，不能修改为代码模式，也不能删除。在客户端请求该服务时，引擎会加载该服务的内容，并编译为 js 代码，然后调用这个服务的 `process` 方法，传入客户端请求对象 `req`（Request 的缩写）和服务器响应对象 `res`（Response 的缩写）。`req` 对象里包含了客户端发送过来的所有请求参数，在后台服务的各个语句里，可以通过 `req.参数名` 的形式从客户端请求对象 `req` 中获取指定参数的值。服务器响应对象 `res` 默认仅提供了一个名为 `code` 的属性，值为 0，表示后台服务执行成功，在后台服务的各个语句里，可以用 `res.参数名` 的形式，将需要返回给客户端的数据保存到 `res` 对象中。在所有操作都执行完毕后，可以通过“终止服务”语句结束当前服务，并把响应结果返回给客户端。

在大多数情况下，通过拖拽相关的语句到默认的 `process` 方法里，然后设置好相关参数，就能实现很多常见的功能，例如数据库增删改查、导入导出、上传下载等。如果业务逻辑特别复杂，则建议对业务逻辑进行拆分，将某些相对独立的操作或者可以复用的语句放到单独的方

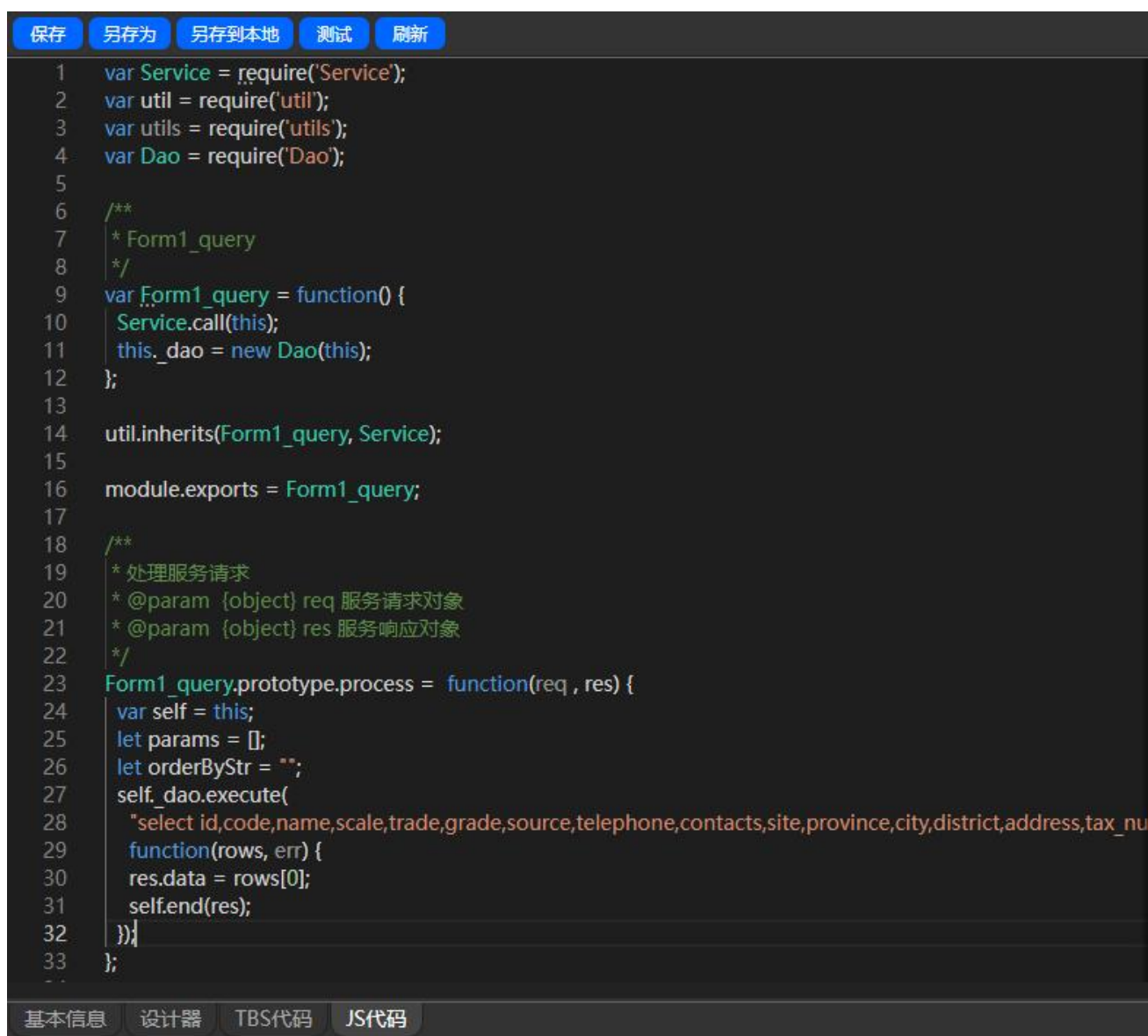
法内,以便查看和维护。新建的方法建议都要求传递 req 和 res 参数,以便在新方法里通过 req 对象获得客户端请求参数,通过 res 对象向客户端返回数据。在一个方法内,可以使用基本语句中的“方法调用”语句调用当前后台服务内的其他方法。

### 3.2.2.3 TBS 代码

点击后台服务设计器底部的“TBS 代码”选项卡,可以查看和编辑当前后台服务的 tbs 源码,但一般不建议直接改 tbs 源码,除非对 tbs 代码的结构很熟悉。

### 3.2.2.4 JS 代码

点击后台服务设计器底部的“JS 代码”选项卡,可以查看编译后的 JavaScript 代码,如下图所示:



```
1  var Service = require('Service');
2  var util = require('util');
3  var utils = require('utils');
4  var Dao = require('Dao');
5
6  /**
7   * Form1_query
8   */
9  var Form1_query = function() {
10     Service.call(this);
11     this._dao = new Dao(this);
12 };
13
14 util.inherits(Form1_query, Service);
15
16 module.exports = Form1_query;
17
18 /**
19  * 处理服务请求
20  * @param {object} req 服务请求对象
21  * @param {object} res 服务响应对象
22  */
23 Form1_query.prototype.process = function(req, res) {
24     var self = this;
25     let params = [];
26     let orderByStr = "";
27     self._dao.execute(
28         "select id,code,name,scale,trade,grade,source,telephone,contacts,site,province,city,district,address,tax_nu
29         function(rows, err) {
30             res.data = rows[0];
31             self.end(res);
32         })
33     };
34 }
```

## 3.3 前端页面文件

在 Web 浏览器、手机 APP 或微信小程序等客户端软件中运行的代码称为前端文件，目前，TaskBuilder 支持开发电脑端网页、手机端 H5 页面和微信小程序，其中，手机端 H5 页面不仅可以用手机上的浏览器打开，还可以嵌入到微信公众号、任讯、阿里钉钉和企业微信中。

### 3.3.1 前端页面的内容格式

TaskBuilder 可以编辑的前端页面类型分为 HTML 和 tfp (Taskmsg Front Page) 两种格式，HTML 格式只能用代码编辑器编辑，这种模式基本很少有人用，大家都用 tfp 格式，因为使用 TaskBuilder 向导创建的前端页面都是采用的 tfp 格式，而且这种格式可以可视化编辑，可以大幅提升开发效率，后续文档所说的前端页面默认都是指 tfp 格式的页面，除非有特定声明。

tfp 文件的实际内容为 JSON 格式，里面包含当前页面的一些设置，包括页面类型、定位方式、终端类型、背景颜色、标题等，然后是各个组件的信息，包括编号、类型、属性、样式、事件等信息，另外，还包括页面的脚本、CSS 样式、数据操作配置等信息。在前端页面设计器内，点击底部的“TFP 源码”选项卡，可以查看当前页面的 tfp 代码，如下图所示：

```
1  {
2      "client": "pc",
3      "framework": "tfp",
4      "positionType": "float",
5      "pageType": "page",
6      "title": "账户管理",
7      "bgColorMode": "light",
8      "type": "Page",
9      "components": [
10         {
11             "id": "panelPage",
12             "type": "Panel",
13             "components": [
14                 {
15                     "id": "panelQuery",
16                     "type": "Panel",
17                     "styles": {
18                         "left": "15px",
19                         "top": "15px",
20                         "right": "15px",
21                         "height": "50px",
22                         "position": "absolute",
23                         "background-color": "#f6f6ff",
```

上面的 JSON 代码是 tfp 页面 **设计时** 的代码，用来保存在 TaskBuilder 里设计该页面时设置的各种信息，以便每次用 TaskBuilder 打开该文件时，能根据文件里的内容生成设计时界面。当有客户端访问 tfp 页面时，引擎会判断在 tasgine\web\dist 目录内是否已经有该文件编译后的 HTML 文件，如果已经存在，而且创建时间晚于 tfp 文件的最后修改时间，则直接将该 HTML 文件的内容返回给客户端，否则，引擎会对该 tfp 文件进行编译，生成对应的 HTML 文件，并将该页面里各种组件所需的 js 和 css 文件的代码打包压缩后合并到该 HTML 文件中，然后将该文件的内容再返回给客户端，客户端（例如 Web 浏览器）进行加载和显示，这时候的代码称为 **运行时** 代码，在运行时，页面里的 js 脚本可以访问页面里的组件并进行控制，实现与用户的交互。

### 3.3.2 前端页面基本信息设置

在用向导创建前端页面时，有以下选项可以选择：

#### 3.3.2.1 终端类型

终端类型是指页面支持的客户端类型，目前 TaskBuilder 支持以下几种类型：

- **电脑端**：可以用电脑端 Web 浏览器访问的页面；
- **手机 H5**：可以用手机端浏览器访问的 HTML5 页面，也可以作为微信公众号的链接页面；
- **任讯 APP**：可以嵌入到任讯 APP 中的 HTML5 页面，开发方式跟手机 H5 页面一样，只不过可以使用任讯 APP 提供的一些扩展功能，包括拍照、扫码、定位等；
- **阿里钉钉**：可以嵌入到阿里钉钉 APP 中的 HTML5 页面，开发方式跟手机 H5 页面一样，只不过可以使用阿里钉钉 APP 提供的一些扩展功能，包括拍照、扫码、定位等；
- **企业微信**：可以嵌入到企业微信 APP 中的 HTML5 页面，开发方式跟手机 H5 页面一样，只不过可以使用企业微信 APP 提供的一些扩展功能，包括拍照、扫码、定位等；

由以上信息可知，TaskBuilder 开发的前端页面总体上分为电脑端和手机端两大类，但根据客户端软件的不同又可以细分为更多类型。

#### 3.3.2.2 页面类型

页面类型是指页面的展现形式，目前 TaskBuilder 支持以下几种类型：

- **普通页面**：指打开后显示在后台管理系统整个功能操作区的页面，主要用于数据列表或内



容较多的页面，多个页面可以通过点击顶部系统标签栏里各个页面的标签进行切换。

- **对话框**：是指在后台管理系统内显示在其他页面上层的弹出对话框页面，默认在整个后台管理系统的界面中居中显示，需要设置固定的宽度和高度，如果页面内容较少，则建议用对话框。
- **抽屉页**：是指从后台管理系统的右侧滑出的页面，默认宽度为 900px，高度为 100%，主要用来作为添加、修改和查看详情的页面。抽屉页在设计时跟普通页面没有什么区别，只是打开方式不一样。

对话框和抽屉页是任擎后台管理系统顶层页面提供的功能，如果将 tfp 页面嵌入到其他系统内，则不支持以对话框或抽屉页的形式显示。

### 3.3.2.3 背景颜色

页面的背景颜色有浅色和深色这两种色系可以选择，我们用 TaskBuilder 开发的应用管理系统一般都是浅色背景，很少有用深色背景的，深色背景目前主要是给 TaskBuilder 自身的一些功能页面用的，TaskBuilder 里有不少页面都是用 TaskBuilder 自己开发的 tfp 页面，例如前后端文件创建向导、前后端各种语句的参数设置对话框等，因为 TaskBuilder 默认是深色背景，所以在这些页面中，相关组件都需要按深色背景进行样式适配，包括组件背景颜色、文字颜色、边框颜色、选中后背景颜色和文字颜色、鼠标滑过时的背景和文字颜色等等。所有 tfp 组件都有适配浅色背景和深色背景的两套 CSS 样式文件，可以根据当前页面的背景颜色设置自动适配。

### 3.3.2.4 定位方式

用 TaskBuilder 新建 tfp 页面时，有以下两种定位方式可以选择：

- **浮动定位**：这是 tfp 页面默认的定位方式，也是网页默认的定位方式，如果选择这种定位方式，则新添加到页面内的组件，默认会从左至右、从上到下显示，如果组件为容器组件，则会占据整行，否则，一行可以显示多个组件。
- **绝对定位**：如果选择这种定位方式，则从组件库拖拽组件到页面上时，鼠标在哪里松开，组件就会放在什么位置，相当于每个组件都有一个相对于父组件的坐标位置，同一个父组件中的各个子组件的位置互相不会影响。

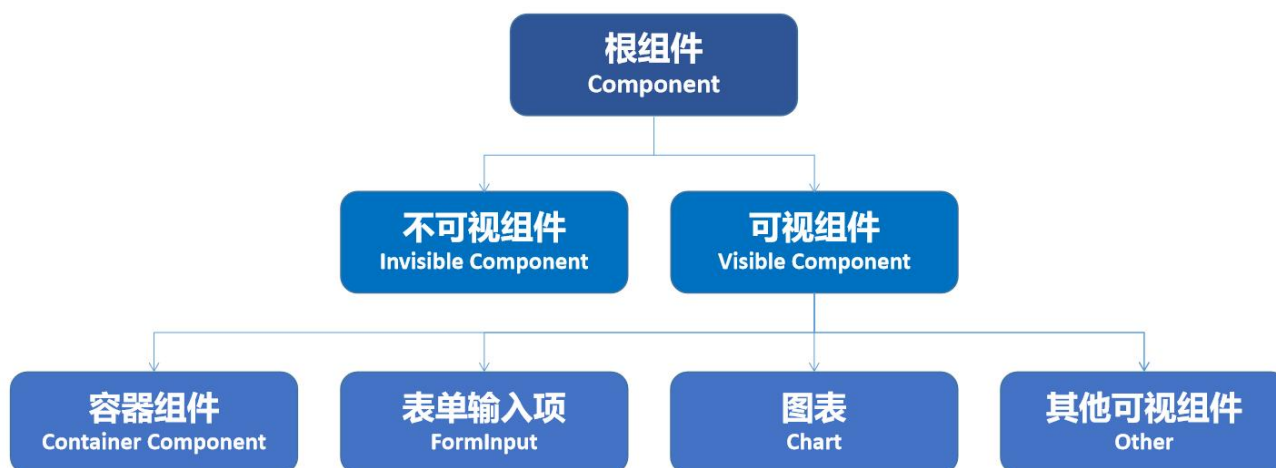
绝对定位虽然可以精确控制各个组件的位置，但如果组件多了后，要实现多个组件的对齐，或者整体位置调整就比较麻烦，需要挨个修改坐标；另外，绝对定位无法根据屏幕尺寸变化而

自动调整大小和位置。所以，我们在开发管理信息系统时，新建的页面都采用默认的浮动定位即可，除非某些特定的页面需要使用绝对定位。

### 3.3.3 前端组件简介

#### 3.3.3.1 前端组件的分类

前端页面是由众多组件层层嵌套构成的，这些组件是任讯信息自主研发的一套前端 UI 组件，称为 tfp 组件，这些组件根据其功能和特点又分为几大类，它们的继承关系如下图所示：



从图中可知，所有组件都是从一个名为 Component 的根组件继承过来的，然后分为不可视组件和可视组件，不可视组件就是在界面里不显示任何内容的组件，例如服务请求组件和计数器组件；可视组件就是在界面里会显示具体内容的组件，分为以下几种类型：

- **容器组件**：是指可以在其内部放置其他组件的容器型组件，例如布局组件、面板组件、表单组件、选项卡组件等。整个 tfp 页面最顶层的组件就是页面（Page）组件，它也是容器组件，其他组件都是放在页面组件或其子组件中的，一个页面有且只能有一个页面组件。
- **表单输入项**：用来显示或录入动态数据的组件，例如单行文本、单选、多选、下拉列表等组件，其中隐藏域组件实际也不在界面里显示任何内容，但是习惯上将其作为输入项放在表单里，用来存放一些临时性的数据，所以没把该组件放到不可视组件中；可以通过 **组件编号.value** 获取或设置输入项的值，通过数据绑定格式属性可以设置输入项在绑定后台数据时采用的格式。
- **图表**：用来显示图形化的统计报表，包括折线图、柱状图、条形图、饼状图等；
- **其他可视组件**：还有一些可视化组件，不属于上面这些类型中的任何一种，包括可编辑表



格、树形组件、文本标签、甘特图等。

### 3.3.3.2 前端组件的三种运行环境

前端组件的代码在整个生命周期中会分别运行在下面三种环境中：

- **设计时**：在 TaskBuilder 的前端页面设计器内对组件进行设计时，组件处于设计时阶段，在该阶段，组件不能进行实际的互动操作，例如不能显示或录入数据，只能用前端页面设计器提供的相关功能对组件的样式、属性和事件等信息进行设置，设置的信息最后都统一按照 JSON 格式保存到扩展名为 tfp 的文件内。
- **编译时**：在用户访问 tfp 文件时，如果该文件还没有编译，或者编译后，tfp 文件又进行了修改，则引擎会将该文件编译为 HTML 格式的文件，然后返回给浏览器，以便浏览器加载显示，在这个阶段，前端组件处于编译时环境，引擎会使用组件的渲染器进行编译。
- **运行时**：当页面在浏览器里显示时，页面中的所有组件就进入了运行时状态，在运行时，组件可以绑定后台服务查询到的数据，可以让用户进行点击、输入、选择等操作。

### 3.3.3.3 前端组件的构成

一个完整的前端组件由以下几部分构成：

#### 3.3.3.3.1 元数据（MetaData）

用来定义每种组件具体都有哪些属性、默认样式、事件和方法，属性包括名称、说明、数据类型、可选值和默认值等，事件包括名称、说明、参数名称和类型等，方法包括名称、说明、参数名称和类型等。元数据相当于组件的数据结构，在 TaskBuilder 里选中组件后，通过元数据可以决定在右侧的属性和事件设置面板里具体要显示哪些属性和事件，在 JS 脚本编辑器里调用组件方法时，也是通过分析组件的元数据来决定具体哪个组件有哪些方法可以调用，以及需要传递哪些参数等，下图是图片组件的元数据内容：

```

1  export default {
2    "name": "Image",
3    "comment": "图片",
4    "attrs": [
5      {
6        "name": "src",
7        "comment": "图片路径",
8        "type": "PathPicker",
9        "fileTypes": ["image"]
10     },
11     {
12       "name": "title",
13       "comment": "图片提示",
14       "type": "String",
15       "isHtmlAttr": true
16     },
17     {
18       "name": "alt",
19       "comment": "替换文本",
20       "type": "String",
21       "isHtmlAttr": true
22     }
23   ],
24   "styles": ["size", "position", "font", "background", "margin", "border", "other"],
25   "defaultStyles": {
26     "width": "48px",
27     "height": "48px"
28   },
29   "events": [
30     {
31       "name": "onAfterInitRuntime",
32       "comment": "组件初始化后",
33       "isNotHtmlEvent": true
34     }, {
35       "name": "onClick",
36       "comment": "点击时"
37     }, {
38       "name": "onMouseOver",
39       "comment": "鼠标经过时"

```

### 3.3.3.2 数据模型（DataModel）

这里的数据模型是指用来存储前端组件的属性、样式和事件等信息的 JSON 对象，与前面介绍的用来存储数据库表或视图信息的数据模型不是一个东西。在 TaskBuilder 里设置组件的样式、属性和事件后，设置的信息都会保存到该组件对应的数据模型对象内，然后在保存 tfp 页面时，会转换为 JSON 格式的字符串保存到 tfp 文件中，在前端页面设计器底部点击“TFP 源码”可以看到页面内所有组件的数据模型的 JSON 格式的信息。在运行时，可以通过 **组件编号.dataModel** 获得组件的数据模型信息，但不建议直接修改该变量的值，因为单独修改数据模

型中的属性或样式值不会造成组件对应的 HTML 元素的变化, 应该通过设置组件控制器对象的属性、样式或调用其相关方法来实现, 因为执行这些操作时, 不仅会修改数据模型里对应的属性或样式值, 还会同步修改组件的相关 HTML 元素。

所有 tfp 组件的数据模型都有以下属性:

- **id**: 组件编号, 类型为字符串, 同一个页面里, 组件编号不能重复;
- **type**: 组件类型, 类型为字符串, 例如 Panel (面板组件)、Label (文本标签)、Button (按钮) 等;
- **styles**: 组件的 CSS 样式设置, 类型为 JSON 对象, 里面的每个属性对应一个 CSS 样式。

如果是容器组件, 则会多一个名为 components 的属性, 类型为数组, 里面是子组件的 JSON 对象。其他属性则根据组件类型的不同而不同, 例如所有输入项组件都有 value 属性, 表示输入项的值, 数据表格组件有 columns 属性, 用来设置表格的列等。

另外, 如果组件设置了事件, 则在组件的数据模型对象中, 还会出现事件类属性。

点击前端页面设计器底部的“TFP 源码”可以查看页面内所有组件的数据模型代码, 下图是一个按钮组件的数据模型代码:

```
{
  "id": "button_add",
  "type": "Button",
  "buttonType": "primary",
  "theme": "rect",
  "value": "添加",
  "labelSize": 14,
  "labelHeight": "30",
  "imageWidth": 16,
  "imageHeight": 16,
  "showImage": true,
  "imageUrl": "tfp-button-icon tfp-button-icon-new",
  "styles": {
    "flex-grow": 0,
    "flex-shrink": 0,
    "flex-basis": "auto",
    "margin-left": "10px"
  },
  "onClick": "button_add_onClick()"
}
```

### 3.3.3.3 控制器（Controller）

我们通常情况下所说的组件实际上就是指该组件的控制器对象，不管是在设计时，还是编译时或运行时，都需要使用控制器对象来设置组件的数据和控制组件的行为。

在运行时，tfp 前端框架会为页面里的每个 tfp 组件生成一个对应的控制器对象，可以用组件编号作为变量名来访问该组件的控制器对象，通过该对象，可以访问或操作组件的以下信息：

- **值**：所有输入项类型的组件都有一个名为 value 的特殊属性，称为组件的值，可以通过 **组件编号.value** 的形式获取或设置组件的值。

获取和设置输入项值的示例代码：

```
let txt = text1.value; //获取 text1 组件的值并赋值给 txt 变量
text1.value = 'Hello'; //将 text1 组件的值设置为 "Hello"
```

- **属性**：是指给每种组件定义的一些参数，在 TaskBuilder 的前端页面设计器里选中组件后，可以在右侧的属性设置面板里查看和设置这些属性。有些属性是通用的，例如所有组件都有组件编号（id）、组件类型（type）这两个属性，所有可视组件都有 CSS 样式类和自定义 CSS 样式属性，所有表单输入项都有数据绑定格式、组件备注、输入提示、必填、只读、禁用等属性。其他属性则根据组件类型的不同而不同。属性基本上都是在设计时设置的，某些组件的某些属性只能在设计时设置，不支持在运行时修改，具体请查看组件的说明。

设置组件属性的示例代码：

```
image1.src = '/images/logo.jpg'; //设置名为 image1 的图片组件要显示的图片的路径
```

- **样式**：是指可以控制组件外观显示的 CSS 样式设置项，例如宽度、高度、文字颜色、背景颜色、边框样式、边距等，在 TaskBuilder 的前端页面设计器里选中组件后，可以在设计器上部的样式设置面板里查看和设置这些样式。不可视组件不需要也不能设置样式。

设置组件样式的代码：

```
label1.css('color', 'red'); //将名为 label1 的文本标签组件的文字颜色设置为红色
```

- **事件**：是指在页面运行的过程中，由浏览器或者用户执行的某个操作触发的、与各个组件有关的事件，例如页面内容加载完毕事件、按钮点击事件、表单提交后事件等。在 TaskBuilder 的前端页面设计器里选中组件后，点击右侧边栏里的“事件”选项卡，可以在此查看或设置组件的各种事件，在事件输入框中双击鼠标可以查看和编辑事件处理函数的具体内容，例如点击确定按钮后提交表单，表单提交完毕后关闭当前对话框等。事件一般不需要通过

脚本手动触发，所以就不提供示例代码了。

- **方法：**是指各种组件对外提供的各项功能，以便在页面脚本中调用，例如调用表单组件的 `submit()` 方法可以提交表单，调用数据表格组件的 `reloadData()` 方法可以重新加载表格数据。获取或设置组件的值、属性和样式，可以调用组件的方法，代用组件方法的示例代码如下：

```
form1.submit(); //提交表单组件 form1 中所有输入项的数据等等
```

#### 3.3.3.3.4 渲染器（Render）

是指用来生成组件 HTML 代码的 JavaScript 对象，可视组件都必须有渲染器，渲染器可以通过组件的数据模型里设置的属性、样式和事件等信息生成该组件对应的 HTML 代码。设计时会使用组件的渲染器动态生成组件的 HTML 代码并显示在页面中，编译时会使用组件的渲染器生成组件的 HTML 代码保存到整个页面的 HTML 文件中，运行时，正常不需要使用组件的渲染器，因为编译时已经把组件的 HTML 代码都生成了，但如果要使用 `tfp.new()` 方法动态创建组件，则也会用到组件的渲染器。

#### 3.3.3.3.5 样式文件

可视组件都有专门的 CSS 样式文件，用来定义组件内各个 HTML 元素的显示样式，有的组件还会针对深色背景和浅色背景提供两套 CSS 样式文件，以便针对不同的背景颜色提供不同的外观样式。

#### 3.3.3.3.6 资源文件

前端组件用到的图片、图标、以及第三方的 CSS 文件和 JS 脚本等统称为组件的资源文件，这些文件属于辅助文件，不会编译打包到最终的 HTML 文件中，

在编译 `tfp` 文件时，会先用各个组件的渲染器根据自己数据模型里的信息生成整个页面的 HTML 代码，然后再将页面里所有组件的元数据、控制器和样式文件的代码全部打包压缩后放到该 HTML 文件中，这样，当浏览器打开该 HTML 页面时，就不需要再从服务器分别加载这些信息，可以实现页面的秒开。

### 3.3.4 前端页面 UI 界面设计

为了方便大家快速设计前端页面的 UI 界面，TaskBuilder 提供了可视化的 UI 界面设计器，



在该设计器内，可以从左侧的组件库中拖拽组件到页面上实现组件的添加，TaskBuilder 提供了几十种丰富的 UI 组件，分为容器组件、只读组件、基础输入项、扩展输入项、不可视组件、导入导出组件、统计图表和媒体组件等，每个组件都可以设置详细的样式、属性和事件，可以实现 UI 界面的高保真设计。支持拖拽组件改变位置，支持剪切复制粘贴组件，支持撤销重做。

tfp 前端页面的 UI 界面设计器如下图所示：



从上图可知，整个 UI 界面设计器从上至下、从左至右分为以下几个区域：

- (1) **工具栏**：在此提供了保存、另存、预览、刷新、复制、剪切、粘贴、删除、撤销、重做、上移、下移等按钮，点击即可执行对应的操作；
- (2) **样式设置栏**：在此可以精确设置组件的各种外观样式，包括显示和定位、字体和文本、边框和边距、背景这几大类；
- (3) **组件库和组件大纲**：组件库里提供了几十款常用 UI 组件，点击左侧的组件分类图标可以分别查看各个类别里的组件列表，将组件库里的组件拖拽到页面上，即可实现组件的添加。点击组件大纲，可以按树形结构查看当前页面里所有组件的列表；
- (4) **组件路径导航**：在此可以查看从当前选中组件一直追溯到页面组件的所有父组件的路径，以便清晰了解当前组件的嵌套关系，点击其中任何一个父组件都可以直接选中该组件；
- (5) **页面内容编辑区**：在此编辑当前页面的内容，可以鼠标拖拽组件的位置，点击组件可以

选中组件，然后在顶部样式栏设置组件样式，在右侧属性和事件栏设置组件的属性和事件；

(6) **不可视组件列表**：用来显示当前页面里不可视的组件，例如服务请求组件和计时器；

(7) **属性和事件设置面板**：在此可以设置组件的属性和事件，每种组件的属性和事件都不一样，默认显示属性列表，点击该面板顶部的“事件”选项卡，可以查看和设置组件的事件；

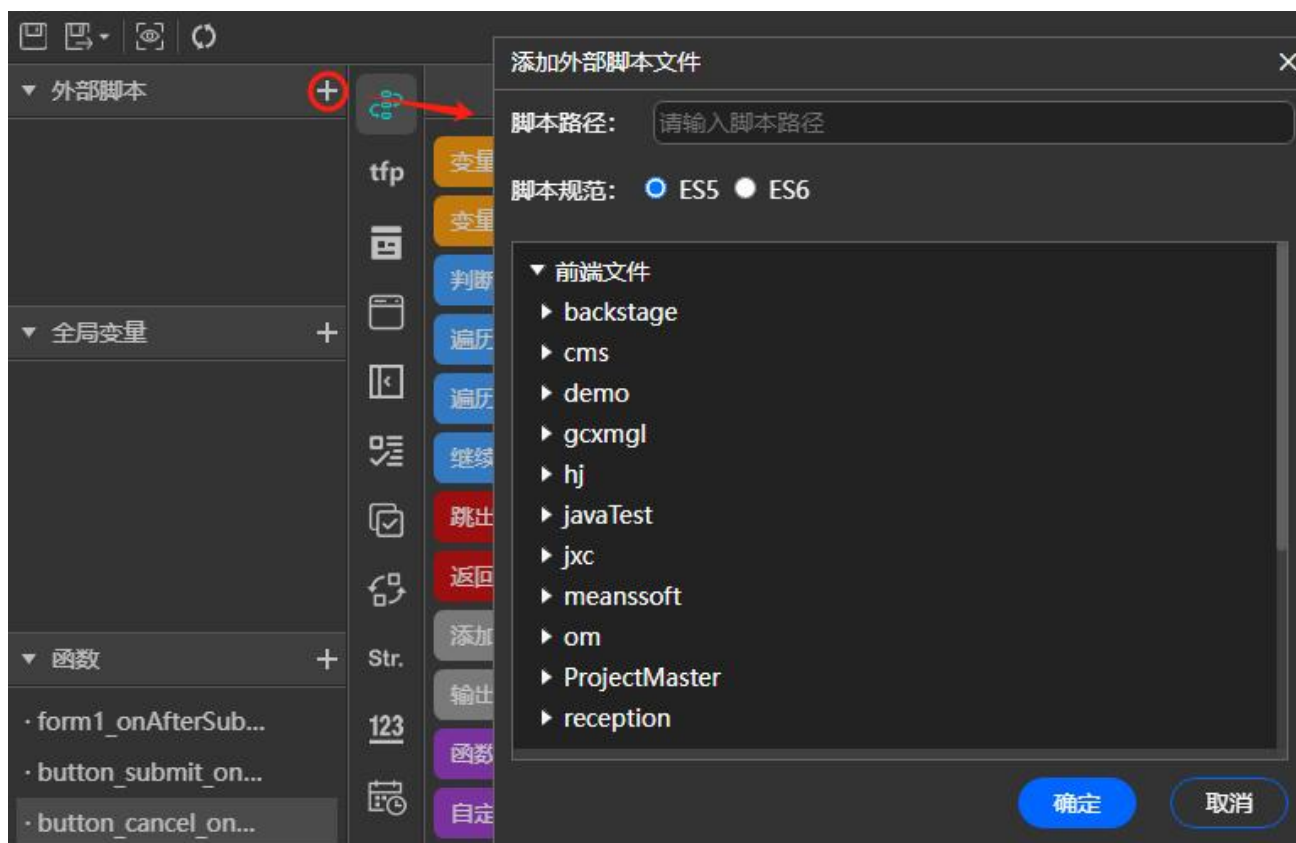
### 3.3.5 前端页面 JS 脚本编辑

在前端页面设计器内，点击底部的“JS 脚本”选项卡，可以打开 JS 脚本编辑器，查看和设置当前页面的 JavaScript 脚本有关的内容，包括以下几个部分：

#### 3.3.5.1 外部脚本引用

如果在前端页面里需要用到外部 js 文件的功能，则可以点击 JS 脚本设计器左上角“外部脚本”栏右侧的加号，即可打开“添加外部脚本文件”的对话框。在该对话框中，可以在下部前端文件列表里选择当前引擎服务器上的前端 js 脚本，也可以直接输入 js 脚本文件的路径，支持引入第三方服务器上的 js 脚本，只需要提供完整的、可以访问的 url 路径即可。

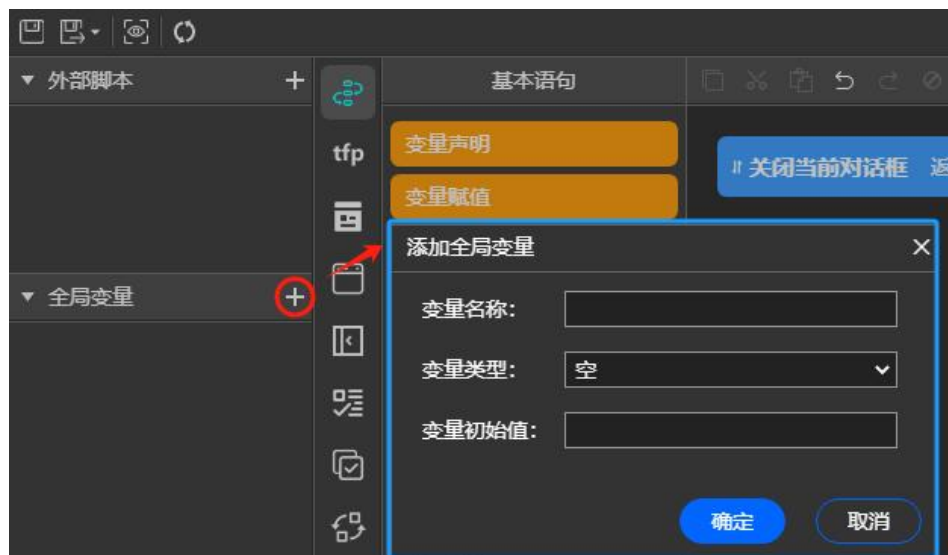
另外，需要选择 js 脚本支持的语言规范的版本，默认为 ES5，如果是 ES6 模块，则需要选择 ES6。该窗口的界面如下图：





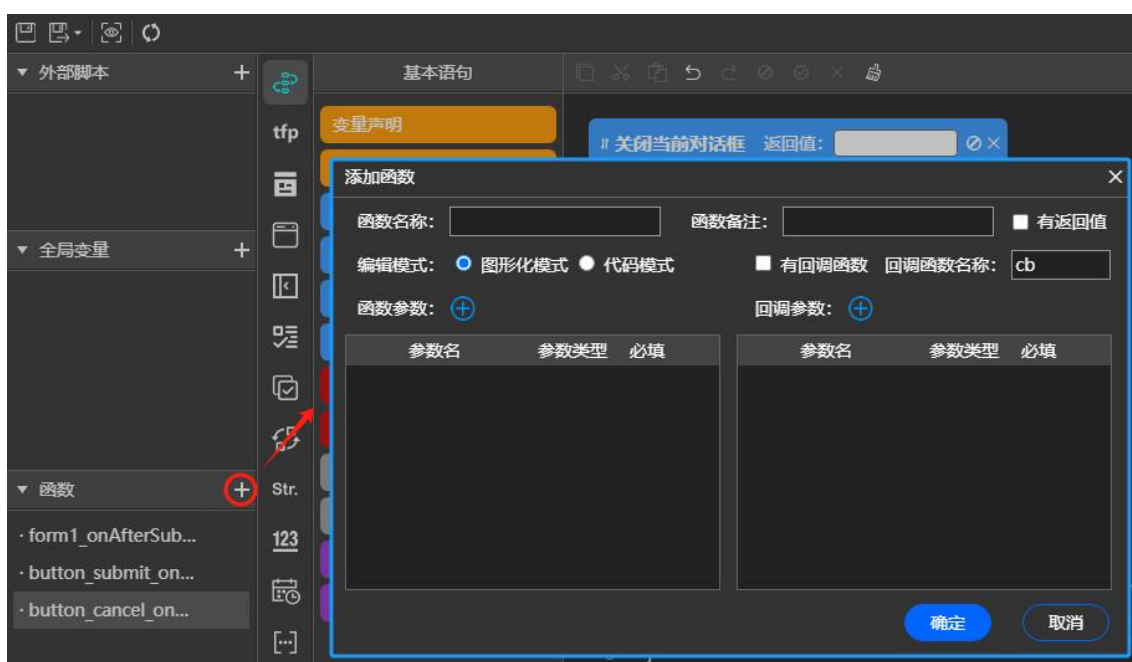
### 3.3.5.2 全局变量

如果前端页面中定义了多个函数，而且需要在多个函数之间共享数据，则可以通过全局变量来实现，点击 JS 脚本设计器左侧“全局变量”栏右侧的加号按钮，即可打开“添加全局变量”的对话框，如下图所示：



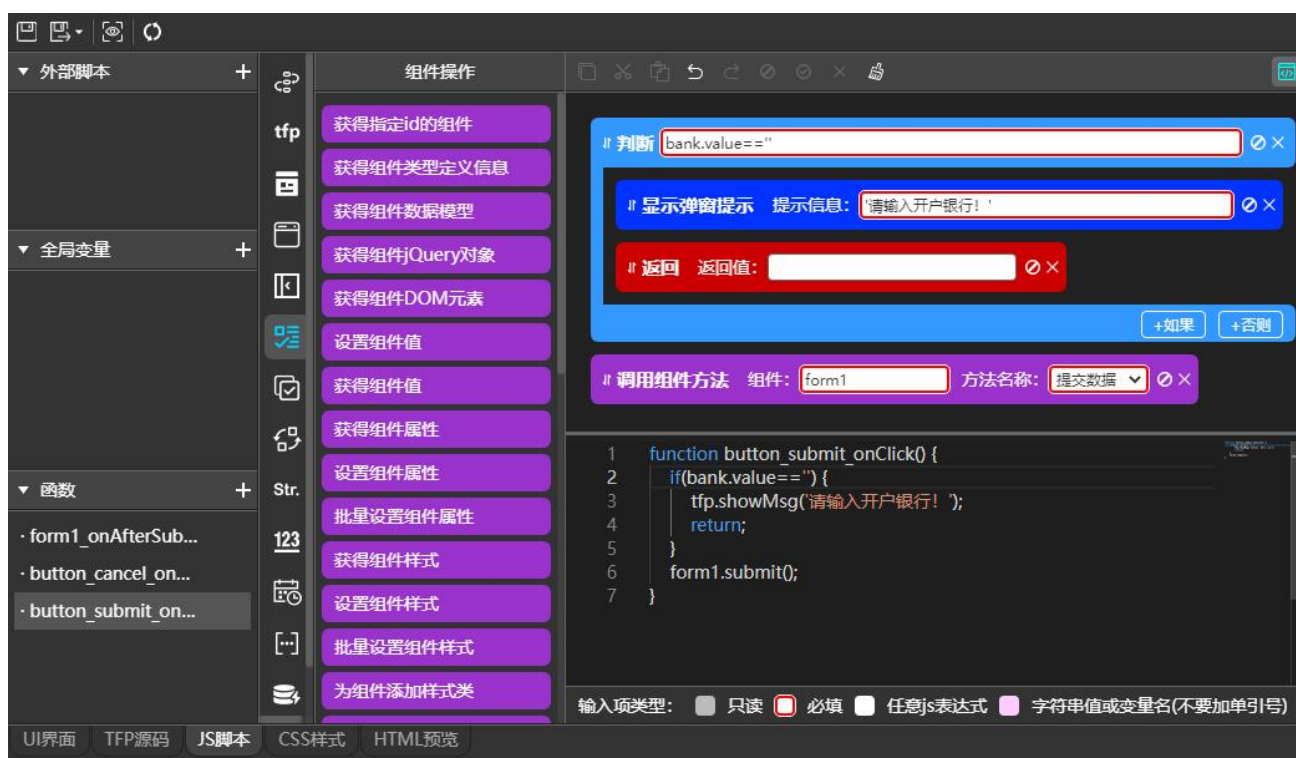
### 3.3.5.3 函数

函数主要用来实现组件事件的业务逻辑，例如点击确定按钮后提交表单里输入项的数据，点击取消按钮关闭当前对话框等，也可以将特定的功能封装到函数中，以便多个函数调用，实现代码的复用。点击 JS 脚本设计器左侧“函数”栏右边的加号按钮，即可打开“添加函数”的对话框，如下图所示：

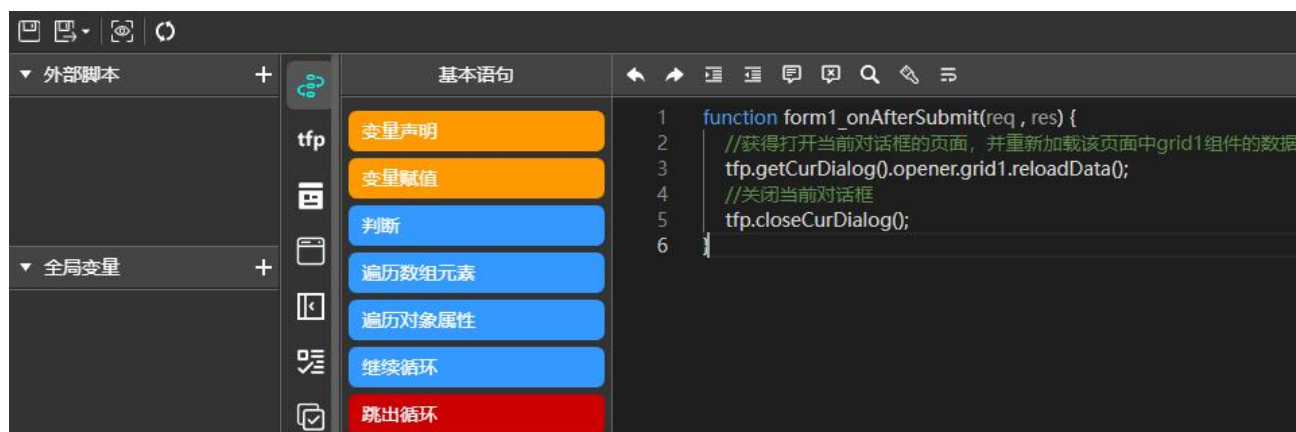


在该对话框中输入函数名称、备注，选择编辑模式，设置好调用该函数需要传入的参数，点确定即可完成函数的创建，如果函数有返回值，则选中“有返回值”复选框，如果有回调函数，则选中“有回调函数”，并可以设置回调函数名称和回调参数。

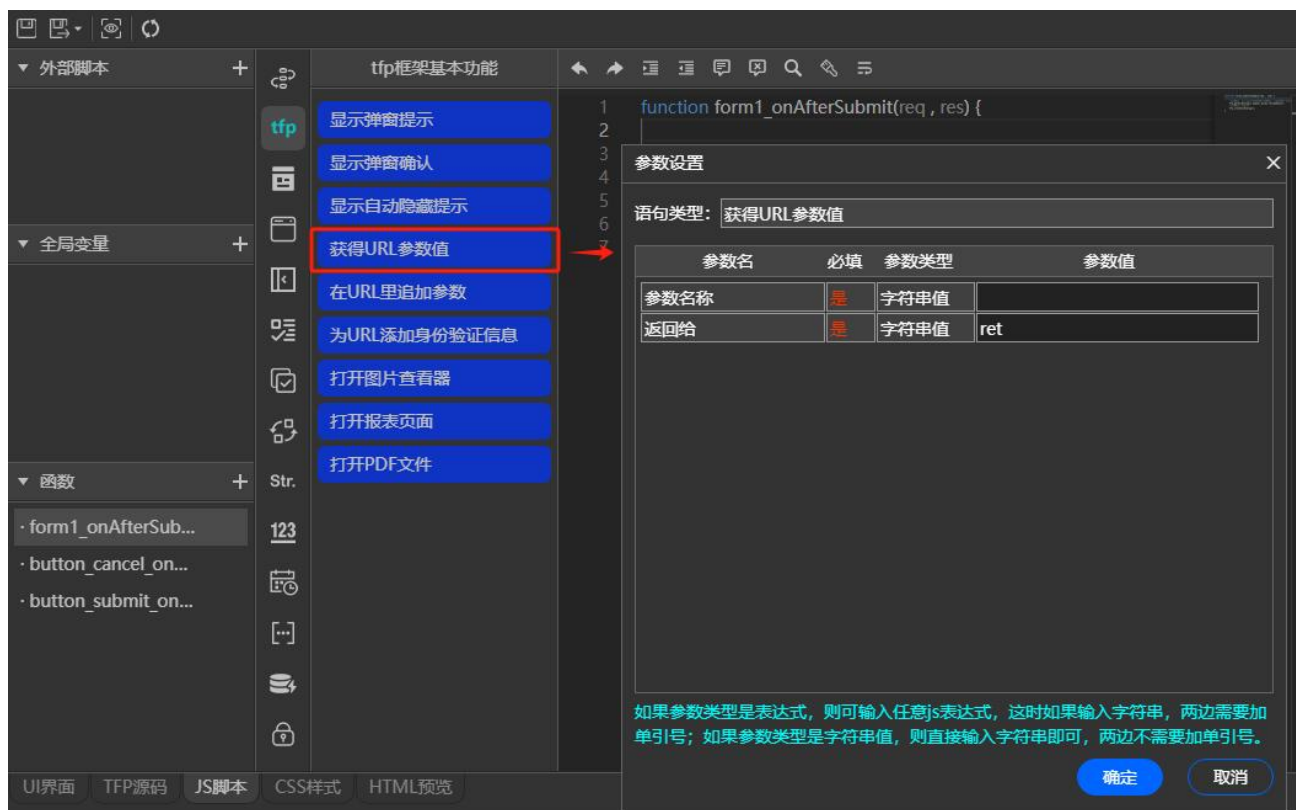
其中，函数的编辑模式支持图形化和代码两种模式，如果是图形化模式，会在设计器右侧显示一个可视化业务逻辑编排区，可以在此编排当前函数的业务逻辑，可以从左侧语句库中拖拽各种语句到右侧可视化业务逻辑编排区，并设置其详细参数，TaskBuilder 提供了一百多个常用的前端操作语句，包括基本语句、tfp 框架基础功能、页面操作、对话框操作、抽屉页操作、组件操作、数据格式校验、数据格式转换、字符串处理、数值处理、日期时间处理、数组处理、浏览器存储和加密解密这几大类，如下图所示：



如果是代码模式，则只能在代码编辑器中编写当前方法的逻辑代码，如下图所示：



当然，也可以从左侧语句库中拖拽语句到代码编辑器，但需要先设置好语句的参数，点确定后，就会将该语句的相应代码插入到光标位置，如下图所示：



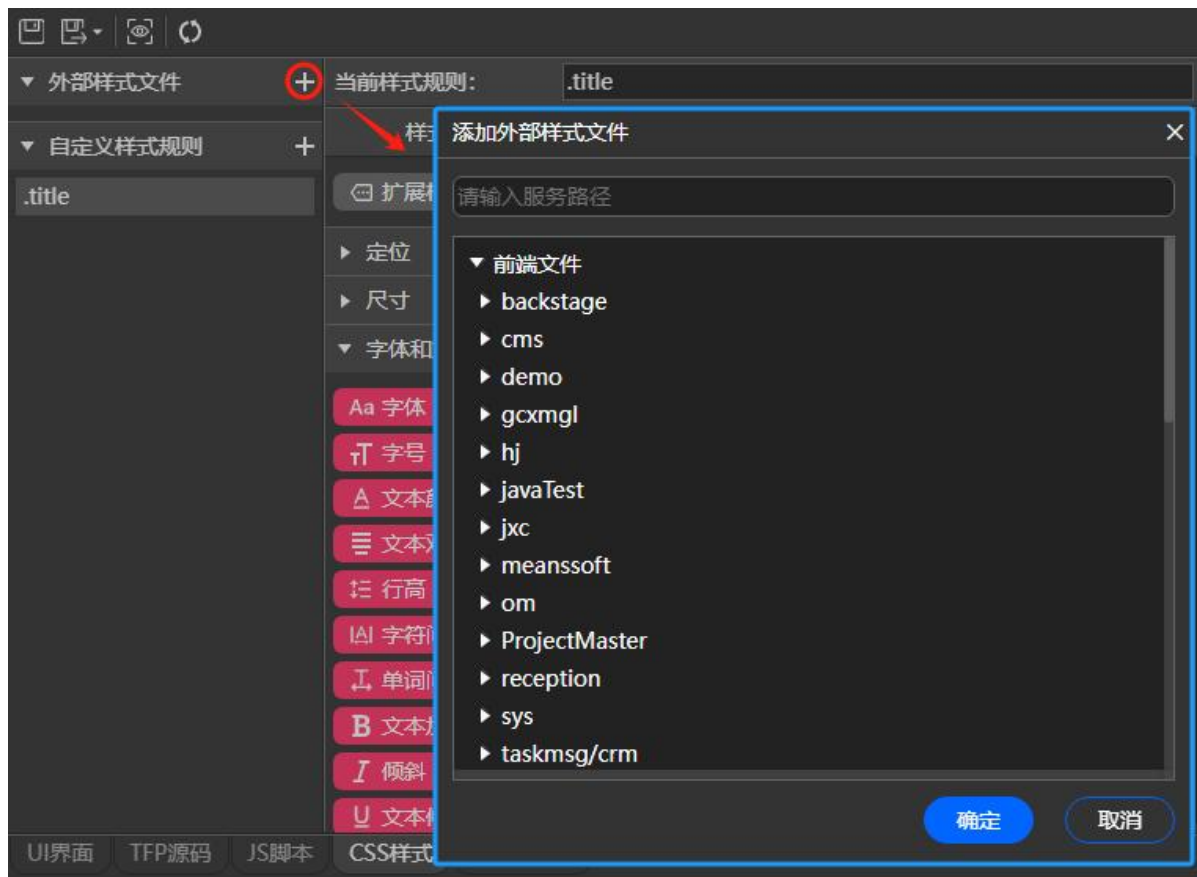
### 3.3.6 前端页面 CSS 样式规则设置

在前端页面设计器内，点击底部的“css 样式”选项卡，可以打开 CSS 样式设计器，在此查看和设计当前页面的 CSS 样式规则，如下图所示：



### 3.3.6.1 引入外部样式文件

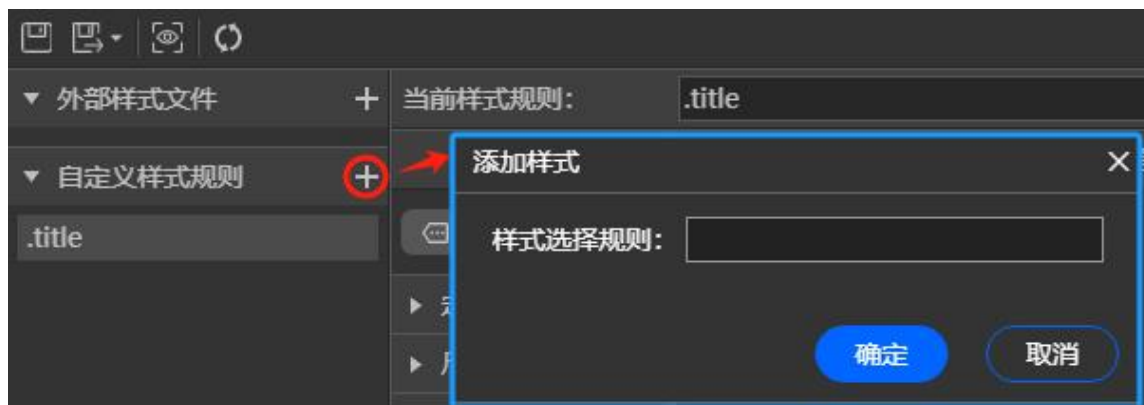
如果要在页面中引入外部 CSS 文件，可以点击 CSS 样式设计器左上角“外部样式文件”栏右边的加号按钮，可以打开“添加外部样式文件”对话框，如下图所示：



在该对话框中，可以从下部前端文件列表里选择当前引擎服务器 web 目录下的 CSS 样式文件，也可以直接输入其他服务器上的 CSS 样式文件的 URL 路径。

### 3.3.6.2 设置自定义样式规则

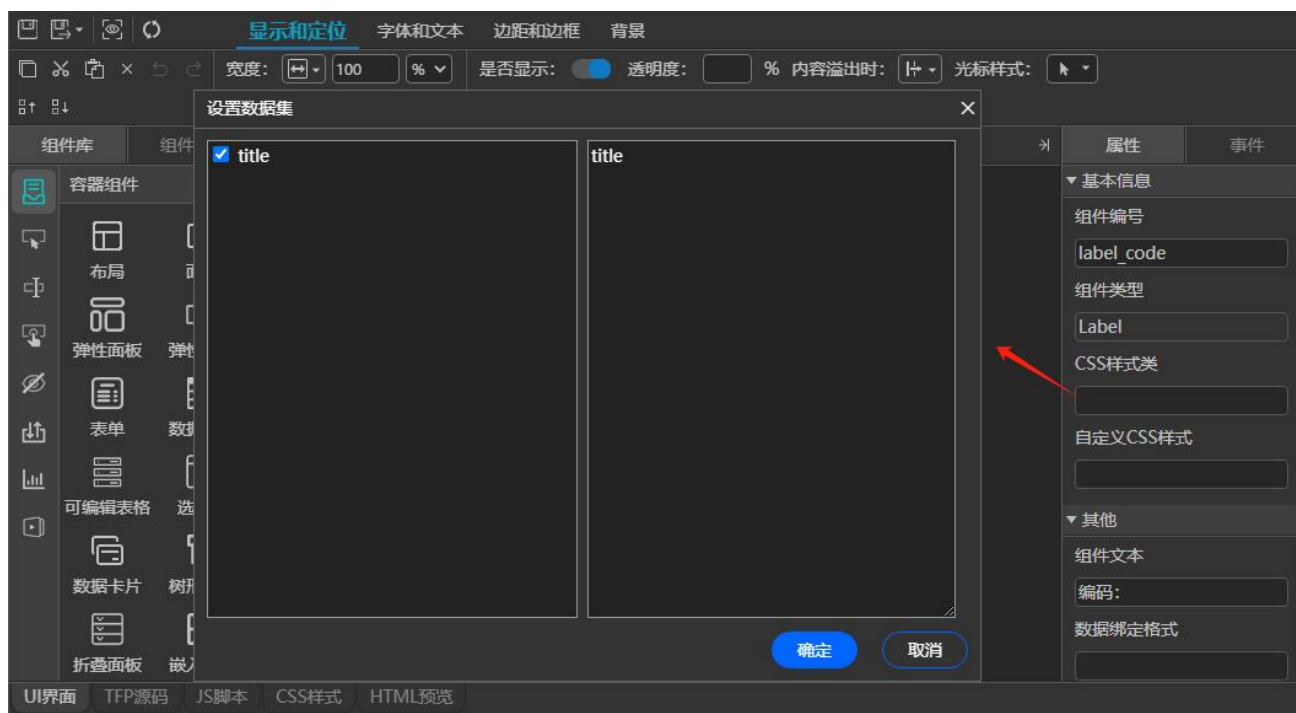
点击 CSS 样式设计器左边“自定义样式规则”栏右边的加号按钮，可以打开“添加样式”对话框，如下图所示：



在该对话框中输入样式选择器的规则，点确定按钮即可创建空白的样式规则，然后从左边样式库中拖拽样式设置项到右边的样式设计区，输入相关参数，即可实现样式规则的设置。

### 3.3.6.3 样式规则的应用

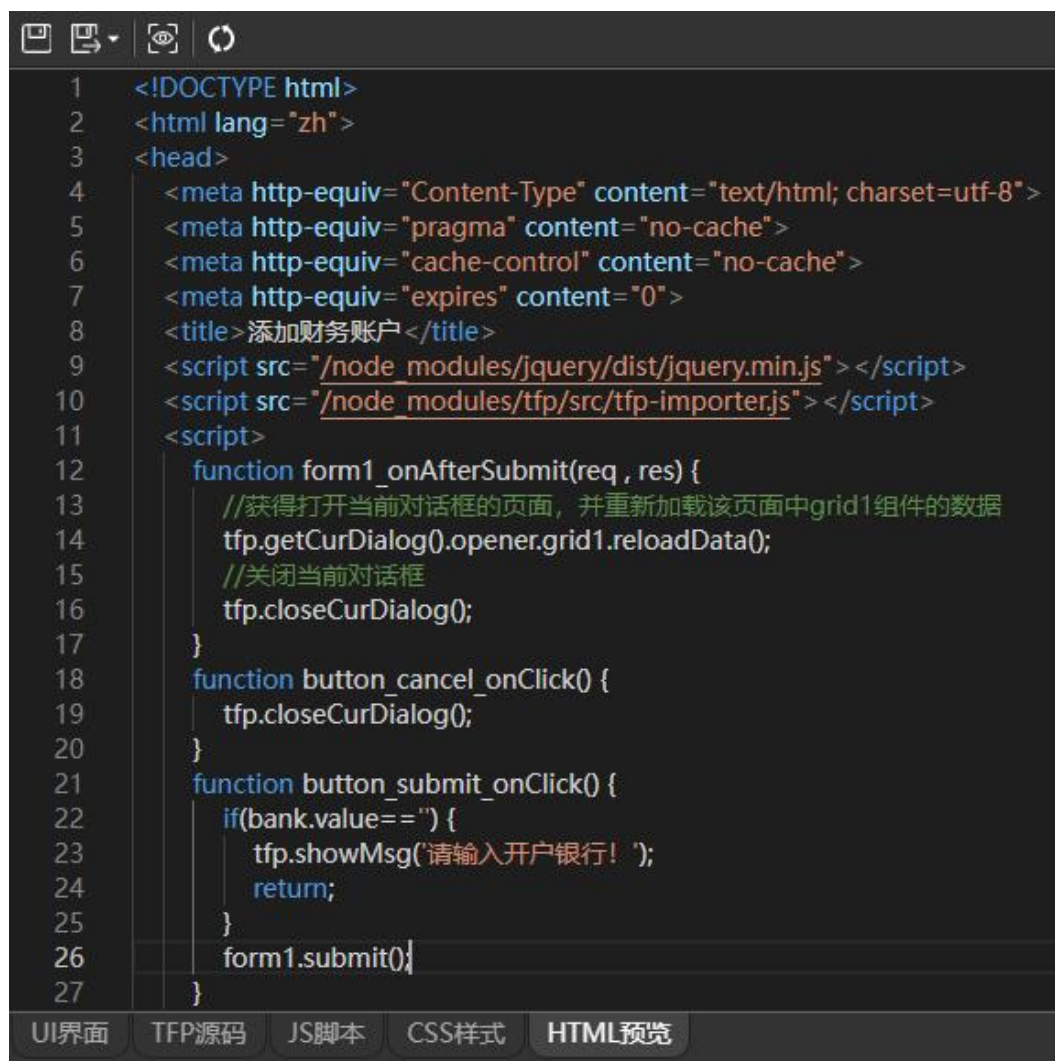
如果设置的样式规则是类选择器（.class-name），则设置好样式规则后，点击前端页面设计器底部的“UI 界面”选项卡，在 UI 界面设计器中选中要应用该规则的组件，然后在属性设置面板的“CSS 样式类”属性的输入框里双击鼠标，打开 CSS 样式类选择对话框，选中要应用的样式类，点确定，即可将该样式规则应用到该组件上，如下图所示：



### 3.3.7 查看前端页面的 HTML 代码

在前端页面设计器内，点击底部的“HTML 预览”选项卡，可以预览当前 tfp 页面编译后的 HTML 代码，但只能查看，不能修改，如下图所示：





```
1 <!DOCTYPE html>
2 <html lang="zh">
3 <head>
4   <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5   <meta http-equiv="pragma" content="no-cache">
6   <meta http-equiv="cache-control" content="no-cache">
7   <meta http-equiv="expires" content="0">
8   <title>添加财务账户</title>
9   <script src="/node_modules/jquery/dist/jquery.min.js"></script>
10  <script src="/node_modules/TFP/src/TFP-importer.js"></script>
11  <script>
12    function form1_onAfterSubmit(req, res) {
13      //获得打开当前对话框的页面，并重新加载该页面中grid1组件的数据
14      tfp.getCurDialog().opener.grid1.reloadData();
15      //关闭当前对话框
16      tfp.closeCurDialog();
17    }
18    function button_cancel_onClick() {
19      tfp.closeCurDialog();
20    }
21    function button_submit_onClick() {
22      if(bank.value=="") {
23        tfp.showMsg('请输入开户银行! ');
24        return;
25      }
26      form1.submit();
27    }

```

UI界面 TFP源码 JS脚本 CSS样式 HTML预览

## 4、前后端数据交互流程解析

### 4.1 客户端请求前端静态文件的流程

任擎服务器支持支持以下三种静态文件：

#### 4.1.1 纯静态文件

纯静态文件是指存放在 tasgine\web 目录或其子目录下，可以直接在浏览器里加载或显示的文件，包括 HTML 文件、CSS 文件、前端 JS 文件、图片、音视频文件等，客户端需要访问这种文件时，向任擎服务器发起请求，并提供这些文件的 URL 地址，任擎就会直接读取这些文件

的内容，然后返回给客户端，客户端再加载和显示。

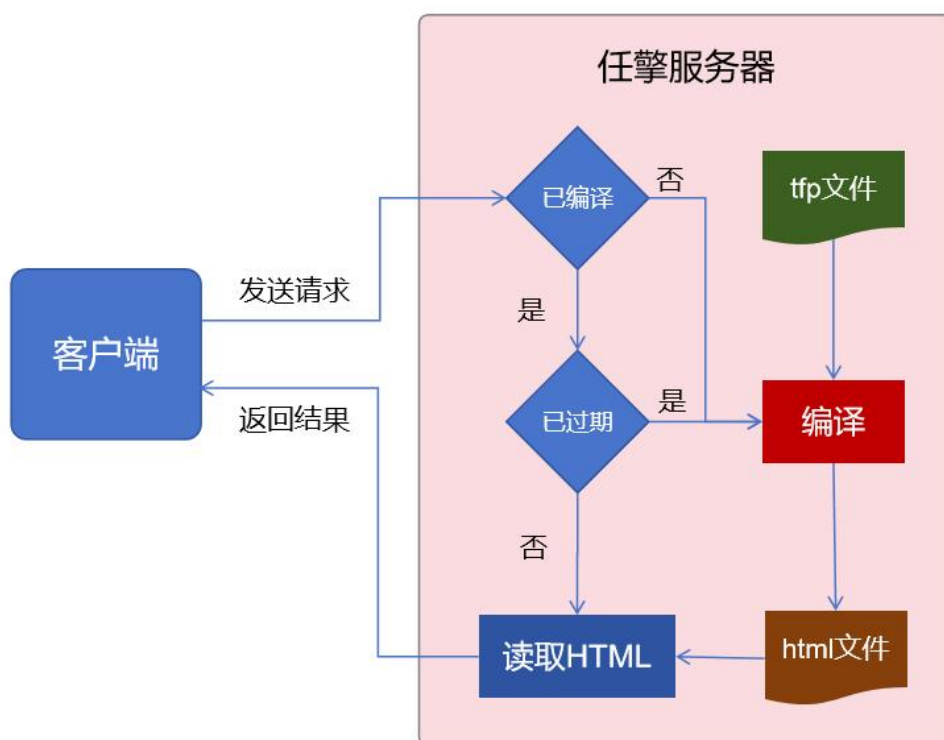
例如任擎服务器的 ip 为 192.168.1.100，监听的网络端口为 8001，如果要访问该服务器上的 tasgine\web\app\demo\index.html 文件，则 URL 地址为：

<http://192.168.1.100:8001/app/demo/index.html>

从上面的路径可知，在 URL 路径里，只需要保留文件实际路径的 tasgine\web 后面的路径即可，但要把斜杠 \ 换成反斜杠 /，任擎默认会从 tasgine\web 目录开始查找静态文件。

### 4.1.2 tfp 文件

扩展名为.tfp 的文件是用 TaskBuilder 设计制作的前端页面文件，文件的内容为 JSON 格式，客户端无法直接解析执行这种格式，需要由任擎服务器先将其编译成 HTML 文件，然后再发送给客户端，客户端才能加载显示，整个请求响应的流程如下图所示：



从上图可知，当有客户端访问 tfp 页面时，任擎会判断在 tasgine\web\dist 目录及其子目录内是否已经有该文件编译后的 HTML 文件，如果已经存在，而且创建时间晚于 tfp 文件的最后



修改时间，则直接将该 HTML 文件的内容返回给客户端。否则，引擎会对该 tpf 文件进行编译，生成对应的 HTML 代码，并将该页面里各种组件所需的 js 和 css 文件的代码都打包压缩后合并到该 HTML 文件中，然后将该文件的内容保存到 tasgine\web\dist 对应的文件夹中，并返回给客户端，客户端再进行加载和显示。

### 4.1.3 附件

如果是通过 tpf 的文件上传或图片上传组件上传的附件文档，则不能通过在 URL 里使用文件路径的方式来直接访问，因为为了保障服务器安全和防止信息泄露，所有附件都是保存在 tasgine\files\组织代码\应用代码\上传年月\ 这样的目录里，而不是放在 tasgine\web 目录里，而且，文件名都变成了 **附件代码.file** 的形式。附件代码是在上传附件时自动生成的一个 32 位的 UUID () 字符串，并且和附件的其他信息一起保存到了 sys\_app\_attach 表中，包括附件名称、类型、大小、保存路径、应用代码、应用数据记录编号、上传人 id、上传时间等。

如果要下载附件，请采用以下形式的 URL：

<http://服务器ip:服务器端口/Download?fileCode=附件代码>

当客户端给引擎服务器发送上述格式的 URL 请求时，引擎会根据 URL 里的附件代码去 sys\_app\_attach 表中查询附件的信息，然后再根据其保存路径读取该附件的实际内容，然后发送给客户端。如果没有找到匹配的附件信息，则返回错误。

如果附件是图片，而且要在浏览器里直接显示，而不是下载，则可以使用下述 URL：

<http://服务器ip:服务器端口/Download?fileCode=附件代码&showImage=true>

## 4.2 前端请求后台服务的方式

在 TaskBuilder 的前端页面设计器的组件库里，有一个名为“**服务请求**”的不可视组件，它是前后端沟通的桥梁，前端页面都是通过这个组件来实现与后台服务的通信，在这个组件的

属性里可以设置具体要请求的后台服务的路径, 可以设置需要传递给后台的请求参数及其类型, 然后其他组件就可以通过这个服务请求组件去访问指定的后台服务进行相关操作, 但大多数情况下不需要手动创建该组件, 因为前端页面内涉及到数据库增删改查的相关组件都提供了数据操作设置, 在这些操作的设置界面里设置好相关参数后, 在运行时, 就会自动创建对应的服务请求组件, 然后就可以通过该组件实现数据库增删改查的操作。

在前端页面里, 具体有哪些方式可以实现对后台服务的请求, 请见下面的介绍:

## 4.2.1 通过组件的数据操作设置自动请求后台服务

在 TaskBuilder 提供的前端组件中, 有不少组件在属性设置中都提供了数据增删改查有关的设置, 例如数据列表组件可以设置数据查询和数据删除操作、表单组件可以设置数据查询和数据提交操作等, 通过这些设置, 组件在运行时就可以自动调用对应的后台服务实现数据的增删改查操作。

在组件的数据操作设置中, 可以选择以下几种方式:

### 4.2.1.1 数据模型

这种方式表示可以直接对指定的数据模型进行增删改查操作, 只需要设置好相关参数, 然后在保存页面时, 就会自动生成该操作对应的 tbs 后台服务文件, 并与该设置的组件进行绑定, 在运行时, 组件就可以请求这个后台服务实现相应的增删改查功能。

自动生成的后台服务的命名规则为:

tfp 文件名(不含.tfp 扩展名)\_组件 id\_操作名.tbs

例如 index.tfp 中的 grid1 组件的数据查询操作生成的后台服务名为:

index\_grid1\_query.tbs

index.tfp 中的 grid1 组件的数据删除操作生成的后台服务名为:

index\_grid1\_delete.tbs

组件的数据查询设置，如果采用直接查询数据模型的方式，则设置界面如下：

数据查询设置

查询数据方式：

数据模型

数据表单

后台服务

SQL语句

HTTP请求

数据模型

请求参数

查询条件

分组和排序

最多查询

条记录

去掉重复记录

不校验是否登录

不检查SQL注入

数据模型路径：

ProjectMaster/model/sale/vpm\_customer.tdm

选择...

添加自定义字段

添加关联字段

|                                     | 字段名称      | 字段备注 | 字段格式(SQL, 双击可编辑) | 字段别名 |
|-------------------------------------|-----------|------|------------------|------|
| <input checked="" type="checkbox"/> | id        | 客户编号 |                  |      |
| <input checked="" type="checkbox"/> | code      | 客户编码 |                  |      |
| <input checked="" type="checkbox"/> | name      | 客户名称 |                  |      |
| <input checked="" type="checkbox"/> | scale     | 客户规模 |                  |      |
| <input checked="" type="checkbox"/> | trade     | 所属行业 |                  |      |
| <input checked="" type="checkbox"/> | grade     | 客户等级 |                  |      |
| <input checked="" type="checkbox"/> | source    | 客户来源 |                  |      |
| <input checked="" type="checkbox"/> | telephone | 联系电话 |                  |      |
| <input checked="" type="checkbox"/> | contacts  | 联系人  |                  |      |
| <input checked="" type="checkbox"/> | site      | 公司网址 |                  |      |
| <input checked="" type="checkbox"/> | province  | 所在省份 |                  |      |

清空

确定

取消

在该对话框中，需要先选择数据模型，然后选择要查询的字段，然后还可以设置请求参数、查询条件、分组和排序条件等参数，点确定，即可完成设置。

组件的数据提交设置，如果采用直接操作数据模型，则界面如下所示：

数据提交设置

提交数据方式：

数据模型

数据表单

后台服务

SQL语句

HTTP请求

数据模型

请求参数

提交类型：

添加

修改

不校验是否登录

不检查SQL注入

数据模型路径：

ProjectMaster/model/finance/pm\_financial\_accounts.tdm

选择...

|                                     | 字段名称          | 字段备注 | 字段值               | 主键                       |
|-------------------------------------|---------------|------|-------------------|--------------------------|
| <input checked="" type="checkbox"/> | code          | 编码   | req.code          | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | name          | 名称   | req.name          | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | type          | 类型   | req.type          | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | bank          | 开户银行 | req.bank          | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | account       | 银行账号 | req.account       | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | remaining_sum | 余额   | req.remaining_sum | <input type="checkbox"/> |
| <input type="checkbox"/>            | status        | 状态   |                   | <input type="checkbox"/> |

清空

确定

取消

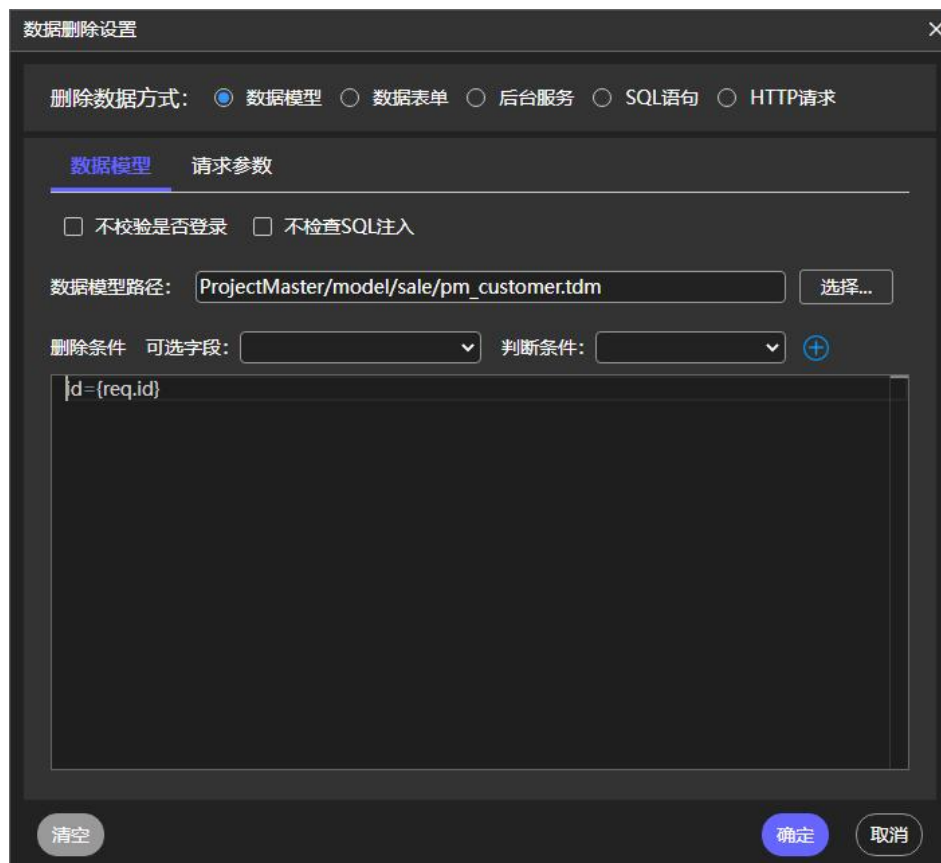
北京任讯信息技术有限公司

160

www.taskbuilder.org

在该对话框中，需要先选择是添加还是修改操作，然后选择数据模型，再选择要操作的字段，并设置好字段的值，然后还可以设置请求参数，点确定，即可完成设置。

在组件的数据删除设置里，如果选择直接删除数据模型里的数据，则界面如下：



在该对话框中，需要先选择数据模型，然后设置删除条件，还可以设置请求参数，最后点确定按钮，即可完成设置。

#### 4.2.1.2 数据表单

这种方式表示可以直接对指定的数据表单的数据进行增删改查操作，实现原理和设置界面与上面的采用数据模型的方式基本一样，设置完后，在保存页面时，也会自动生成对应的 tbs 后台服务，区别就是这里操作的是数据表单自动生成的数据模型，而且，如果一个数据表单里有多数据模型（一个主表，一个或多个子表），则在设置时，可以选择具体要对哪个数据模型进行操作。

数据查询设置采用数据表单的设置界面如下图所示：

数据查询设置

查询数据方式：☐ 数据模型 ☒ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据表单

请求参数

查询条件

分组和排序

最多查询  条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入

数据表单路径:   模型:

| <input checked="" type="checkbox"/> | 字段名称      | 字段备注 | 字段格式(SQL, 双击可编辑) | 字段别名 |
|-------------------------------------|-----------|------|------------------|------|
| <input checked="" type="checkbox"/> | id        | 编号   |                  |      |
| <input checked="" type="checkbox"/> | name      | 名称   |                  |      |
| <input checked="" type="checkbox"/> | code      | 代码   |                  |      |
| <input checked="" type="checkbox"/> | switch3   | 开关   |                  |      |
| <input checked="" type="checkbox"/> | textarea4 | 多行输入 |                  |      |
| <input checked="" type="checkbox"/> | select5   | 下拉框  |                  |      |
| <input checked="" type="checkbox"/> | switch6   | 开关   |                  |      |
| <input checked="" type="checkbox"/> | radio7    | 单选框  |                  |      |
| <input checked="" type="checkbox"/> | tb_status | 状态   |                  |      |
| <input checked="" type="checkbox"/> | tb_ctime  | 创建时间 |                  |      |
| <input checked="" type="checkbox"/> | tb_cusoid | 创建人  |                  |      |

数据提交操作采用数据表单的设置界面如下图所示：

数据提交设置

提交数据方式：☐ 数据模型 ☒ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据表单

请求参数

提交类型: ☒ 添加 ☐ 修改 ☐ 不校验是否登录 ☐ 不检查SQL注入

数据表单路径:   模型:

| <input checked="" type="checkbox"/> | 字段名称      | 字段备注 | 字段值        | 主键                       |
|-------------------------------------|-----------|------|------------|--------------------------|
| <input checked="" type="checkbox"/> | name      | 名称   | req.name   | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | code      | 代码   | req.code   | <input type="checkbox"/> |
| <input type="checkbox"/>            | switch3   | 开关   |            | <input type="checkbox"/> |
| <input type="checkbox"/>            | textarea4 | 多行输入 |            | <input type="checkbox"/> |
| <input type="checkbox"/>            | select5   | 下拉框  |            | <input type="checkbox"/> |
| <input type="checkbox"/>            | switch6   | 开关   |            | <input type="checkbox"/> |
| <input type="checkbox"/>            | radio7    | 单选框  |            | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | tb_status | 状态   | 0          | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | tb_ctime  | 创建时间 | new Date() | <input type="checkbox"/> |

数据删除操作采用数据表单的设置界面如下图所示：

数据删除设置

删除数据方式：☐ 数据模型 ☒ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据表单** 请求参数

☐ 不校验是否登录 ☐ 不检查SQL注入

数据表单路径：

模型： 字段： 判断条件：

#### 4.2.1.3 后台服务

如果组件要使用现有的后台服务或者手动创建的后台服务实现数据增删改查操作，则可以在组件的数据增删改查设置里，选择“后台服务”这种方式，界面如下图所示：

数据查询设置

查询数据方式：☐ 数据模型 ☐ 数据表单 ☒ 后台服务 ☐ SQL语句 ☐ HTTP请求

**后台服务** 请求参数

☒ 直接从后台选择指定的后台服务

☐ 使用页面上已有的后台服务组件

查询结果参数名：

从上图可知，有两种方式可以使用现有的后台服务：

(1) **直接从后台选择指定的后台服务**：这种方式只需要提供一个后台服务的路径即可，可以弹窗选择，也可以手动输入；这种方式还可以设置请求参数。

(2) **使用页面上已有的服务请求组件**：这种方式需要先拖拽一个服务请求组件到页面里，并设置好后台服务的路径以及其他相关参数后，再在这里选择，这么做的好处是多个组件可以共用一个服务请求组件，不需要重复设置。

这种方式一是可以利用现有已经创建好的后台服务，不需要重复创建，二是如果后台业务逻辑很复杂，使用前面那种自动创建的后台服务无法满足需求，就需要使用手动创建的后台服务，可以在里面按需设置复杂的业务逻辑，例如涉及多个数据模型的增删改查操作。

#### 4.2.1.4 SQL 语句

对于 SQL 比较熟悉的同学，如果想直接写 SQL 语句来实现数据的增删改查操作，则在设置组件的数据操作时，就可以选择“SQL 语句”这种类型，例如执行存储过程，设置界面如下：

数据查询设置

查询数据方式：☐ 数据模型 ☐ 数据表单 ☐ 后台服务 ☒ SQL 语句 ☐ HTTP 请求

**SQL 语句** 请求参数

数据源： ☐ 不校验是否登录 ☐ 不检查SQL注入

[使用SQL助手编辑](#)

[清空](#) [确定](#) [取消](#)



在上图对话框的 SQL 编辑器中，可以输入要执行的 SQL 语句，在 SQL 语句中可以以 {req. 参数名} 的形式插入客户端请求参数，点击“请求参数”选项卡，可以设置客户端具体会传哪些参数给后台服务，以及这些参数的类型。

如果采用该方式，在保存页面时，会自动生成执行该 SQL 语句的后台服务，并与该设置所属的组件进行绑定，运行时，组件就可以自动请求该服务实现相关功能。

#### 4.2.1.5 HTTP 请求

如果要通过第三方系统提供的基于 HTTP 协议的 API 接口实现数据的增删改查操作，则在组件的数据操作设置里，可以选择“HTTP 请求”这种类型，设置界面如下图：

The screenshot shows a '数据查询设置' (Data Query Settings) dialog box with a close button (X) in the top right corner. The '查询数据方式' (Query Data Method) section at the top has five radio buttons: '数据模型' (Data Model), '数据表单' (Data Form), '后台服务' (Backend Service), 'SQL 语句' (SQL Statement), and 'HTTP 请求' (HTTP Request), with 'HTTP 请求' selected. Below this, the 'HTTP 请求' (HTTP Request) tab is active, showing a '请求参数' (Request Parameters) sub-tab. The configuration fields include: '请求地址' (Request Address) as a text input; '请求类型' (Request Type) with radio buttons for 'GET' and 'POST' (selected); 'POST 数据类型' (POST Data Type) as a dropdown menu showing 'application/json'; and '查询结果参数名' (Query Result Parameter Name) as a text input showing 'data'. There are also expandable sections for 'HTTP 头' (HTTP Header) and 'Cookie', each with a plus icon and an empty table with columns '名称' (Name) and '值' (Value). At the bottom, there is a '请求数据' (Request Data) section with a large text area, and three buttons: '清空' (Clear), '确定' (Confirm), and '取消' (Cancel).

在上图的对话框中，需要设置第三方系统 API 接口的 URL 地址、支持的请求类型（默认为 POST）、POST 的数据类型、HTTP 头、Cookie 参数、请求数据等，另外，还可以设置请求参数。

如果采用该方式，在保存页面时，会自动生成向第三方系统发送 HTTP 请求的后台服务，并与该设置所属的组件进行绑定，运行时，组件就可以自动请求该服务实现相关功能。

### 4.2.2 通过 JavaScript 脚本手动请求后台服务

在某些复杂的业务中，上面介绍的那种通过组件的数据操作设置来请求后台服务的方式可能无法满足要求，例如需要满足某些条件或者触发某些事件后才能请求后台服务，或者请求后台服务与界面组件无关，只是查询一些数据用来做判断，则这些情况下，就可以在页面中新建一个服务请求组件，设置好服务路径和请求参数后，再使用 JavaScript 脚本手动发起请求。

具体操作步骤如下：

- (1) 从组件库中拖拽服务请求组件到页面上，然后设置好后台服务地址以及请求参数等；
- (2) 在相关组件的事件里编写 JavaScript 脚本，调用上面的服务请求组件的 request() 发送请求；
- (3) 在 request() 方法的回调函数里解析服务器端返回的数据，然后进行相关处理；

调用服务请求组件的 request() 方法发送请求的示例代码如下：

```
//动态设置请求参数
service1.setArg('arg1', '123');

//发送请求
service1.request(null, function(req, res) {
    //将后台服务返回的数据绑定到数据表格组件中
    grid1.bindData(res.data);
});
```

## 4.3 触发前端组件请求后台服务的常见事件

当以下事件发生时，可能会触发相关前端组件向后台服务发起请求：

### 4.3.1 页面加载时

这种情况主要用于数据查询，在前端页面加载完毕后（页面静态内容渲染完毕），会自动判断页面里的每个前端组件是否设置了数据查询操作，如果设置了，则自动请求相应的后台服务进行数据查询，并将查询结果绑定到组件内，例如数据表格组件、表单组件、下拉列表组件、数据卡片组件、树形组件等，但数据表格组件如果在属性里设置了“不自动加载数据”，则不会自动加载，需要用脚本手动加载数据，主要是用于一些特殊场景，例如页面里通过选项卡组件放置了多个数据表格，除了第一个选项卡的数据表格需要自动加载，其他选项卡的数据表格可以在点击对应的选项卡后再通过脚本手动加载，这样可以加快页面打开速度，不用每次都加载所有选项卡中的列表数据。

### 4.3.2 表单提交时

这种情况主要用于数据添加和修改，在表单组件里放置各种输入项，然后设置好表单的数据提交参数，点击确定按钮时，调用表单组件的 `submit()` 方法，就可以将表单里各个输入项的数据提交给对应的后台服务，实现数据添加或修改服务。

### 4.3.3 点击删除按钮时

数据表格组件和树形组件都提供了数据删除的功能，在这两个组件的属性里设置好数据删除有关的参数后，就可以实现点击删除按钮时，删除指定行或节点的数据，具体的设置方法后面会详细介绍。

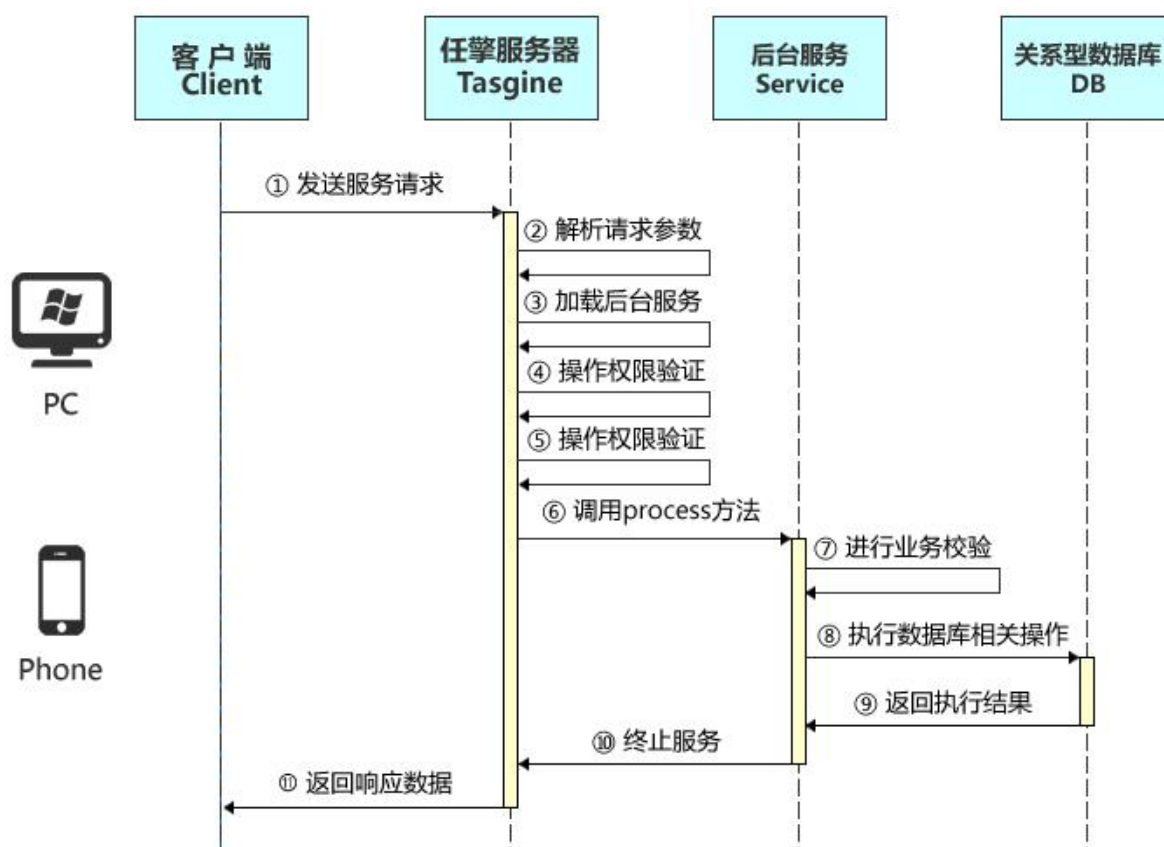
### 4.3.4 关联输入项的值发生变化时

如果前端页面里的服务请求组件设置的请求参数中，有类型为“组件值”的参数，并且服务请求组件启用了“参数变化时自动重载”，则当请求参数里设置的这些组件的值发生变化时，

就会触发该服务请求组件重新发起请求。例如数据列表页面，数据表格组件的数据查询设置里，请求参数关联了页面上的查询条件输入项，则当这些输入项的值发生变化时，就会将这些查询条件的值发送给后台服务进行重新查询，然后在列表里显示最新的查询结果。

## 4.4 客户端请求后台服务的流程

前面介绍了很多关于前端页面和后台服务的知识，那前端和后台到底是怎么通信的呢，下面给大家详细介绍一下，整体流程如下图所示：



上图中各个环节的详细说明如下：

- 1、前端页面通过“服务请求”组件向任擎服务器发送服务请求，指定要请求的服务路径，并提供相关的请求参数；
- 2、任擎服务器先对请求参数进行解析，然后将这些参数封装到一个名为 req（Request 的简写，表示客户端请求）的对象中；

- 3、任擎服务器根据客户端请求的服务路径加载对应的后台服务文件，如果是 tbs 格式，则先编译为 js 格式，然后创建对应的后台服务对象；
- 4、任擎服务器判断该服务是否需要登录校验，如果需要，则根据客户端请求参数中发送的身份验证信息进行登录校验，如果校验失败，则向客户端返回错误信息，并结束该服务的执行；如果校验成功，则将当前用户的相关信息保存到 req.session（会话对象）中，包括当前用户 ID、姓名、代码等，以便在后台服务中引用这些信息；
- 5、如果该服务需要进行登录校验，并且已经通过，则任擎服务器会继续判断该服务是否需要校验操作权限，如果当前用户没有该服务的操作权限，则向客户端返回错误信息，并结束服务的执行；
- 6、上述校验都通过后，任擎服务器会创建一个名为 res（Response 的简写，表示服务器端响应，里面默认就一个属性：code，值默认为 0）的对象，并调用该服务对象的 process(req, res) 方法，并将前面创建的客户端请求对象 req 和服务器端响应对象 res 作为参数传入；
- 7、在服务对象的 process 方法内进行必要的业务校验，例如必填参数是否为空、参数类型是否正确、参数值是否重复等；
- 8、然后执行数据库增删改查、数据导入导出等业务操作；
- 9、然后将处理结果或响应数据保存到 res 对象中；
- 10、最后调用服务对象的 end 方法终止服务的执行，断开数据库连接；
- 11、任擎服务器将 res 对象 序列化为 JSON 字符串返回给客户端。

## 4.5 前后端通讯的数据格式

用 TaskBuilder 开发的应用，前后端传输的数据都是 JSON 格式的。

### 4.5.1 前端页面向后台服务发送的数据格式

前端页面通过“服务请求”组件发送给任擎后台服务的数据是 JSON 格式的，示例如下：

```
{
  "service": "sys/service/service123", //后台服务路径
  "参数 1": "abc", //字符串类型的参数
  "参数 2": 123, //数值类型参数
  "参数 3": ["a", "b", "c"], //数组类型参数
  "参数 4": { //对象类型参数
    "子参数 1": "a",
    "子参数 2": "b",
    "子参数 3": "c", ...
  }
  ...
}
```

### 4.5.2 后台服务返回给前端页面的数据格式

在任擎的后台服务中，可以通过 res 对象（Response 的简称）返回执行的结果，res 对象是 JSON 格式的，每个属性对应一个响应参数，下面是示例及格式说明：

```
{
  "code": 0, //响应代码，默认为 0，表示执行成功，如果不为 0，则表示执行失败
  "message": "错误描述", //如果上面的 code 不等于 0,则该参数会包含具体的错误描述,
  否则没有该参数
  "参数 1": "abc", //字符串类型的参数
  "参数 2": 123, //数值类型参数
  "参数 3": ["a", "b", "c"], //数组类型参数
  "参数 4": { //对象类型参数
    "子参数 1": "a",
    "子参数 2": "b",
    "子参数 3": "c",
    ...
  }
  ...
}
```

## 4.6 常见页面前后端数据交互原理解析

TaskBuilder 主要用来开发数据库增删改查的应用，下面以第五章里创建的那个学员管理项目为例，介绍一下在常见的页面里怎么实现前后端数据交互。

### 4.6.1 数据列表页面前后端交互原理解析

数据列表页面主要通过数据表格组件与后台服务进行通讯，在数据表格组件的属性中，可以通过“数据查询设置”和“数据删除设置”实现列表数据的查询和删除，现在以学员管理项目里的 index.tfp 页面为例，给大家详细介绍一下，请先在 TaskBuilder 里打开学员管理项目前端文件里的 index.tfp，然后选中页面里的数据表格组件 grid1，然后再按下面的文档执行操作。

#### 4.6.1.1 数据查询实现原理解析

##### (1) 设置数据查询参数

点击数据表格组件 grid1 属性设置里的“数据查询设置”，可以看到如下所示的对话框：



数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 查询条件 分组和排序

最多查询  条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

| <input type="checkbox"/>            | 字段名称  | 字段备注  | 字段格式(SQL, 双击可编辑) | 字段别名 |                                  |
|-------------------------------------|-------|-------|------------------|------|----------------------------------|
| <input checked="" type="checkbox"/> | id    | 编号    | id               |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | pxnf  | 培训年份  | pxnf             |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | qs    | 期数    | qs               |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | bc    | 班次    | bc               |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | xm    | 姓名    | xm               |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | xb    | 性别    | xb               |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | sfzh  | 身份证号  | sfzh             |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | whcd  | 文化程度  | whcd             |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | xz    | 乡镇    | xz               |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | xzc   | 行政村   | xzc              |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | sfzkh | 是否贫困户 | sfzkh            |      | <input type="button" value="⊖"/> |



在该对话框中，可以选择查询数据的方式，支持数据模型、数据表单、后台服务、SQL 语句和 HTTP 请求等多种方式，具体区别见前面的介绍。默认采用“数据模型”的方式，需要先选择需要查询的数据模型，选择好后，就会在下面显示该模型中的字段列表，然后可以在该列表中勾选需要查询的字段，如果要关联其他表查询数据，可以点“添加关联字段”设置需要关联的字段信息。

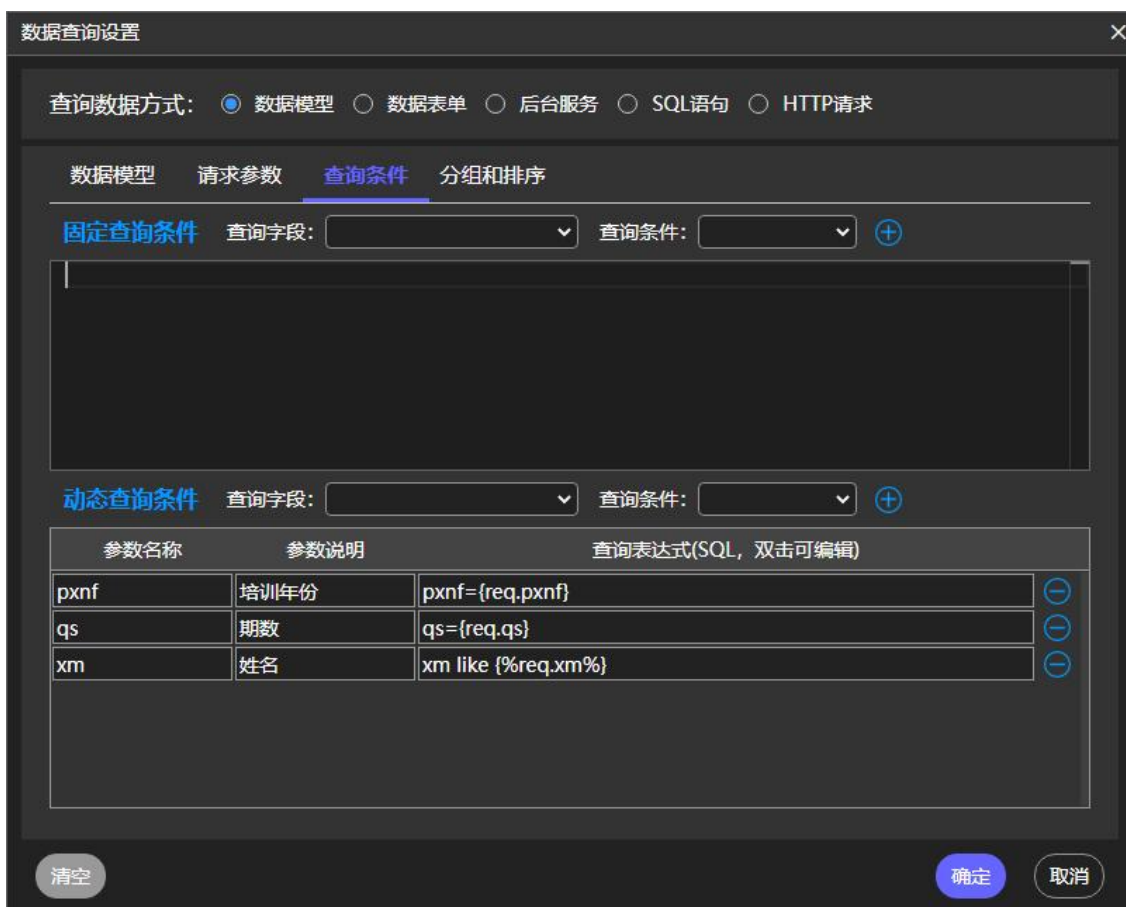
另外，点击该对话框里的“请求参数”选项卡，可以设置页面里哪些输入项的值可以作为请求参数发给后台服务，设置界面如下图所示：



上图的请求参数设置，表示在查询表格数据时，会将页面里编号为 pxnf、qs、xm 的这三个组件的值发送给后台服务，以便在查询条件里作为查询参数使用。

点击“查询条件”选项卡，可以在此设置在后台服务里查询数据时使用的具体查询条件，包括固定查询条件和动态查询条件，固定查询条件就是指在每次查询数据时都会使用的查询条

件,而动态查询条件是指当客户端传递了对应的参数时就使用该条件查询,如果没有传该参数,则不用该条件查询, 设置界面如下图所示:



数据查询设置

查询数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 查询条件 分组和排序

固定查询条件 查询字段:  查询条件:  +

动态查询条件 查询字段:  查询条件:  +

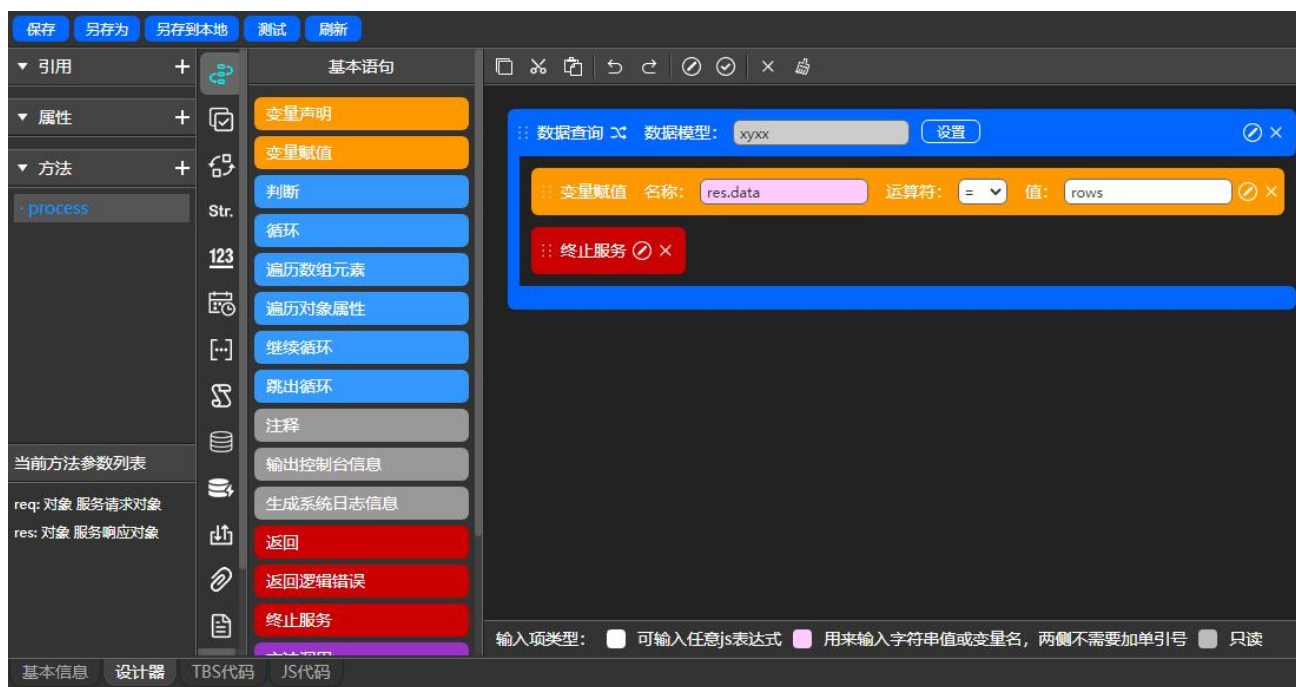
| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑)  |   |
|------|------|--------------------|---|
| pxnf | 培训年份 | pxnf={req.pxnf}    | - |
| qs   | 期数   | qs={req.qs}        | - |
| xm   | 姓名   | xm like {%req.xm%} | - |

清空 确定 取消

从上图可以看到, 动态查询条件里的参数名称必须和前面请求参数里设置的参数名保持一致。在固定查询条件和动态查询条件的表达式里 (SQL 查询语句的 WHERE 条件), 可以用 **{req. 参数名}** 的形式插入参数值, 如果要模糊查询, 则参数值可以使用 **{%req.参数名%}**, 多个动态查询条件是 **与** 的关系, 即有几个条件的参数有值, 则这几个条件就必须都满足。

以上参数正常是在使用 **前端资源创建向导** 创建该页面时就自动配置好的, 并根据这些参数生成了对应的后台服务文件: **index\_grid1\_query.tbs**。如果后面修改了这些参数, 在保存该页面时, 就会自动重新生成对应的后台服务文件, 覆盖原来的文件。

在学员管理项目的后台服务里点击 **index\_grid1\_query.tbs** 文件 (如果看不到, 请在学员管理项目的后台服务节点上点鼠标右键, 然后点刷新), 会在后台服务设计器里打开, 界面如下:



从上图可以看到，在该后台服务中添加了一个数据查询语句，并在其回调函数里面将查询结果保存到 res 对象中，然后终止服务，将查询结果返回给客户端。

点击数据查询语句的设置按钮，可以打开数据查询设置对话框，界面如下：



默认显示的基本信息，包括数据模型、翻页设置、查询数量、重复设置和查询结果参数等。

点击 **查询字段** 选项卡，可以查看具体查询了哪些字段及字段的详细设置，界面如下：

数据查询设置

基本信息

查询字段

动态查询参数

固定查询条件

排序条件

添加自定义字段

添加关联字段

|                                     | 字段名称 | 字段备注 | 字段格式(SQL, 双击可编辑) | 字段别名 | 汇总 |
|-------------------------------------|------|------|------------------|------|----|
| <input checked="" type="checkbox"/> | id   | 编号   | id               |      | ▼  |
| <input checked="" type="checkbox"/> | pxnf | 培训年份 | pxnf             |      | ▼  |
| <input checked="" type="checkbox"/> | qs   | 期数   | qs               |      | ▼  |
| <input checked="" type="checkbox"/> | bc   | 班次   | bc               |      | ▼  |
| <input checked="" type="checkbox"/> | xm   | 姓名   | xm               |      | ▼  |
| <input checked="" type="checkbox"/> | xb   | 性别   | xb               |      | ▼  |
| <input checked="" type="checkbox"/> | sfzh | 身份证号 | sfzh             |      | ▼  |
| <input checked="" type="checkbox"/> | whcd | 文化程度 | whcd             |      | ▼  |
| <input checked="" type="checkbox"/> | xz   | 乡镇   | xz               |      | ▼  |
| <input checked="" type="checkbox"/> | xzc  | 行政村  | xzc              |      | ▼  |

确定

取消

点击 **动态查询参数** 选项卡，可以查看该查询具体设置了哪些动态查询条件，界面如下：

数据查询设置

基本信息

查询字段

动态查询参数

固定查询条件

排序条件

查询字段:  查询条件: 

+

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑)  |              |
|------|------|--------------------|--------------|
| pxnf | 培训年份 | pxnf={req.pxnf}    | <div>⊖</div> |
| qs   | 期数   | qs={req.qs}        | <div>⊖</div> |
| xm   | 姓名   | xm like {%req.xm%} | <div>⊖</div> |

说明：此处设置的查询参数如果前端页面没有传值，则不会作为查询条件。

确定

取消

点击 **固定查询条件** 选项卡，可以查看该查询具体设置了哪些固定查询条件，界面如下：



在创建学员管理项目的数据查询页面时，并没有设置固定的查询条件，所以这里是空的。

点击 **排序条件** 选项卡，可以查看该查询具体设置了哪些排序条件，界面如下：



这里的设置都是在创建或保存页面时自动生成的，请不要在此手动修改。

## (2) 设置数据表格的列

点击数据表格组件 grid1 属性设置里的“数据列设置”，可以打开“数据列设置”对话框，点击该对话框左下角的加号按钮，可以添加列，在左侧列表里选中指定的列，在中间列表里可以选择该列的参数分类，包括基本信息、文本设置（根据内容类型动态变化，例如如果是按钮，则会变成按钮设置）、排序设置、显示条件和行转列设置等，在右侧设置详细参数，界面如下：



其中，最重要的一个参数就是“数据绑定表达式”，在此可以设置在该列里要显示的内容格式，可以用 **{字段名}** 的形式绑定在上面第（1）步里查询的任意字段，可以多个字段组合，或者和固定的字符串值组合，例如：**{pxnf}年第{qs}**，如果查询结果里当前行的 pxnf 字段值为“2024”，qs 字段的值为“2 期”，则这一行这一列最终显示的内容为：**2024 年第 2 期**。

上面的数据绑定设置也可以用 **#[js 表达式]** 的形式实现，这种形式能实现更复杂的功能，操作步骤如下：

先在页面的 JS 脚本里创建一个名为 **getInfo(pxnf, qs)** 的函数，函数的代码如下：

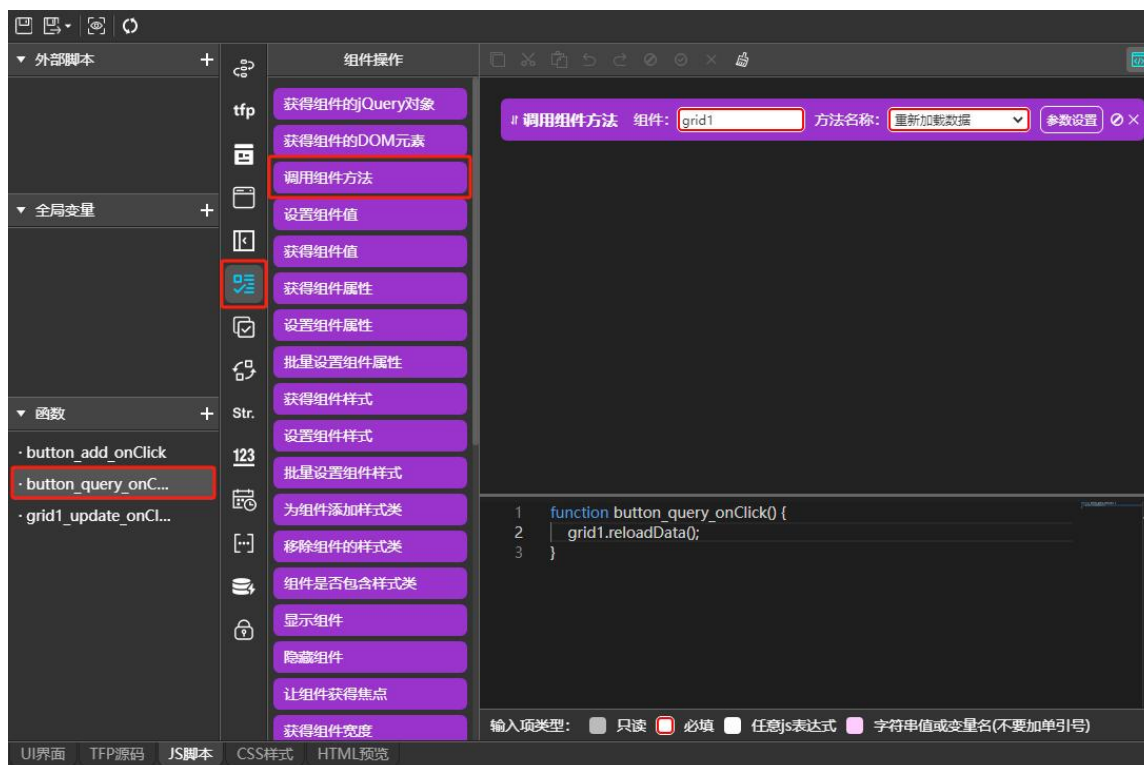


```
function getInfo(pxf, qs) {  
    return pxf+'年第'+qs; //根据传入的 pxf 和 qs 参数，返回拼接后的字符串  
}
```

然后将该列的数据绑定表达式设置为：`#[ getInfo({pxnf}, '{qs}') ]`，这样设置后，在进行该列的数据绑定时，就会调用 `getInfo(pxf, qs)` 这个函数，并将当前行的 `pxnf` 和 `qs` 这两个字段的值传入，然后将返回的值绑定到该单元格中，其中 `qs` 字段的值里包含“期”字，不是纯数值，所以在绑定表达式里，这个字段的大括号外面还需要加单引号，否则会执行报错。

上面只是使用 `#[js 表达式]` 这种形式实现的一个简单的示例，实际可以根据需求实现更复杂的功能，例如数据库里存储的性别值是 0 和 1，0 表示男，1 表示女，则在列表里，就可以创建一个函数，传入性别值，然后在函数里判断，如果性别值等于 0，就返回字符串：“男”，否则返回字符串：“女”，然后性别这一列的数据绑定格式就可以用 `#[ 函数名( {性别字段名} )]` 这种形式实现根据不同的值显示不同的内容。当然，这种情况也可以使用条件显示实现，后面介绍数据表格组件时会详细介绍。

### (3) 设置查询按钮的事件



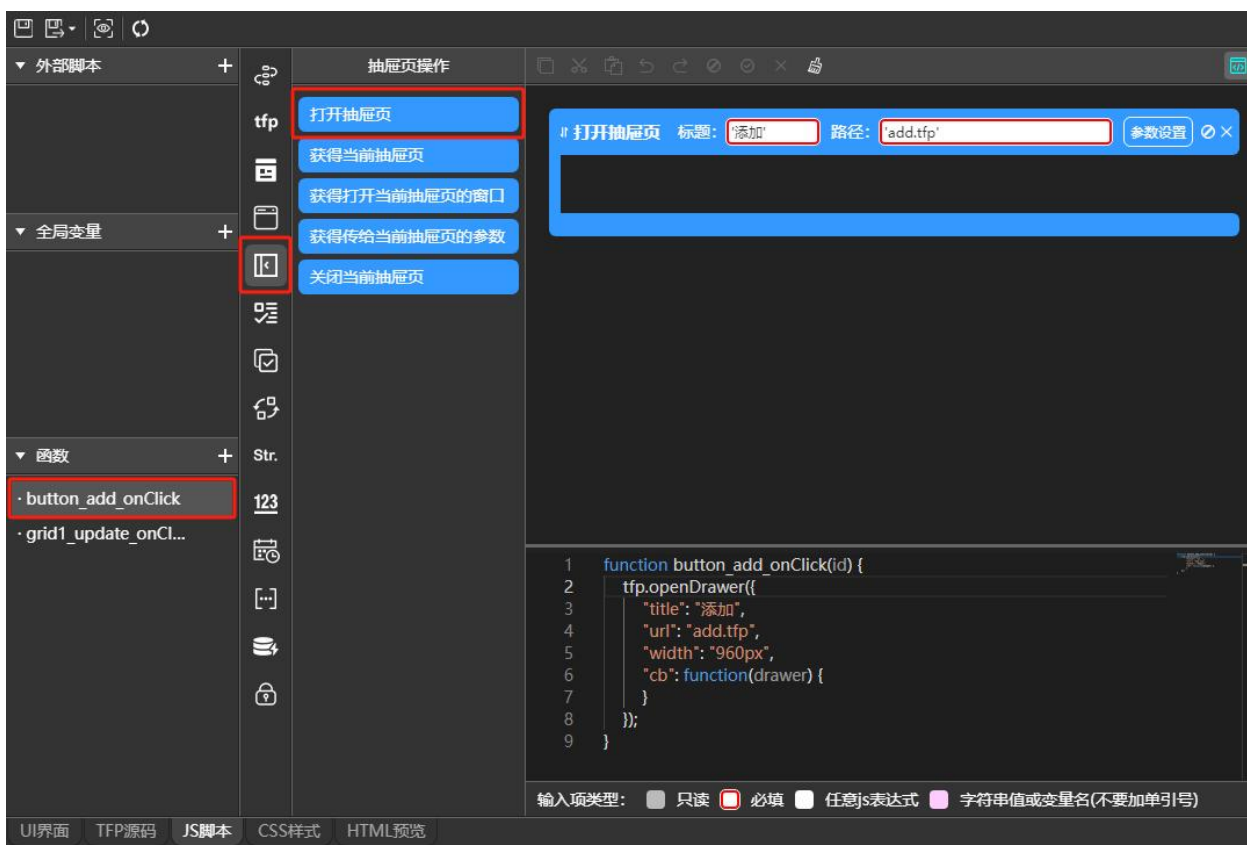


在 index.tfp 页面里选中页面顶部面板里的 **查询** 按钮，点击右侧的 **事件** 设置选项卡，双击 **点击时** 事件输入框，就会看到如上所示的脚本设置界面，里面的这条语句在前端语句库的 **组件操作** 内，意思就是当用户点击 **查询** 按钮时，调用数据表格组件 grid1 的 reloadData() 方法重新加载数据，该方法执行时，会向查询当前表格数据的后台服务重新发起请求，并自动判断页面顶部面板里的培训年份（组件编号为 pxnf）、期数（组件编号为 qs）和姓名（组件编号为 xm）这几个输入项是否有值，如果有就作为请求参数发送给后台，然后将查询结果绑定到数据表格中。

#### (4) 请求后台服务查询并显示数据

在 TaskBuilder 里预览页面或者在浏览器里打开页面时，等页面加载完所有静态内容并解析渲染完毕后，数据表格组件会自动请求数据查询设置里对应的后台服务查询表格的数据，并将查询结果绑定到表格中。

#### 4.6.1.2 打开添加页面实现原理

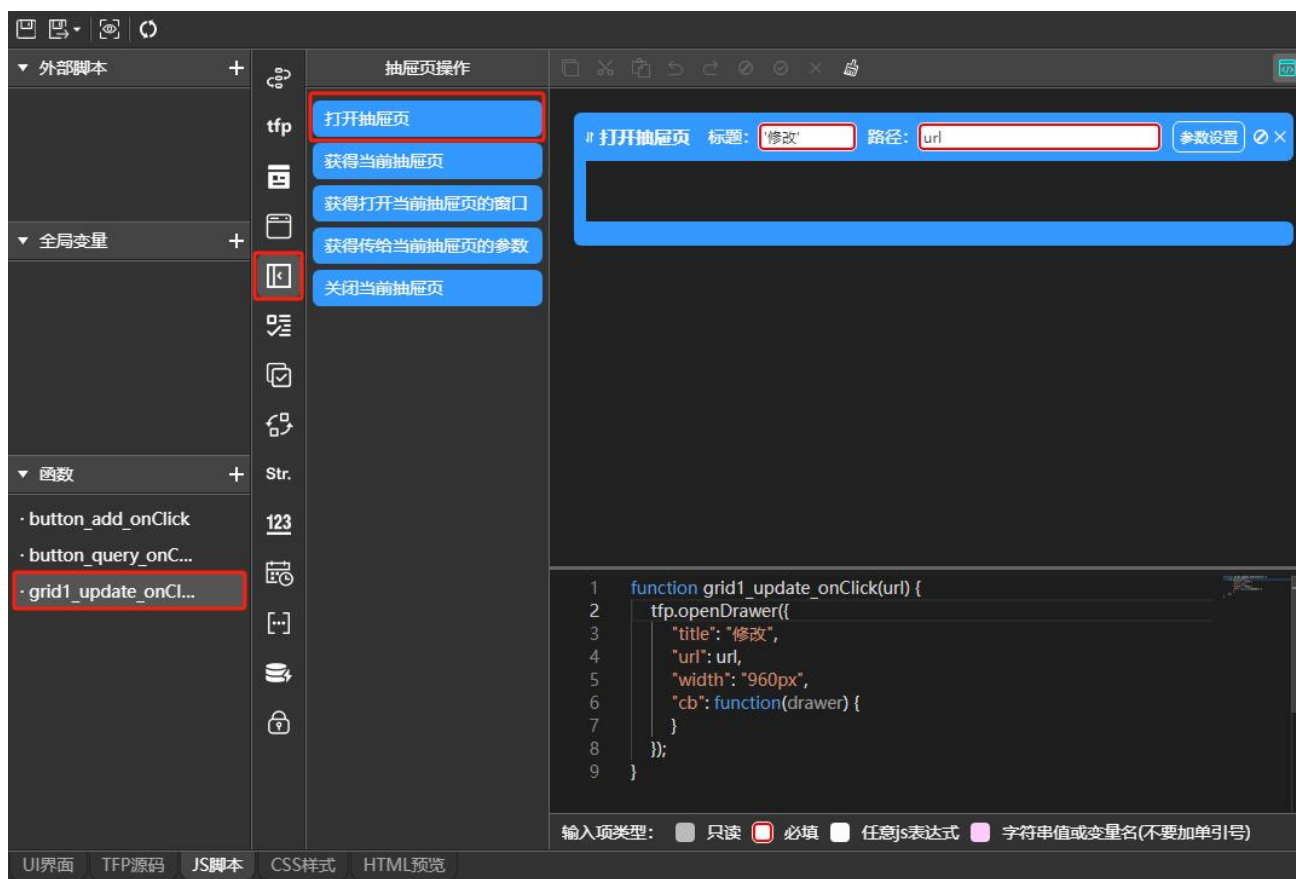


在 index.tfp 页面里选中顶部面板里的 **添加** 按钮, 点击右侧的 **事件** 设置选项卡, 双击 **点击时** 事件输入框, 会看到如上所示的脚本设置界面, 从上图可知, 点击添加按钮后, 会执行打开抽屉页的操作, 该语句在前端语句库的 **抽屉页操作** 里, 在该语句的设置面板里可以设置抽屉页的标题和路径, 这里添加页面的路径默认为 “add.tfp”。

#### 4.6.1.3 打开修改页面实现原理

打开修改页面的设置要稍微复杂一些, 需要先设置好脚本函数, 然后再去数据表格的数据列设置里设置修改这一列的参数, 具体操作步骤如下:

点击前端页面设计器底部的 **JS 脚本** 选项卡, 在左侧函数列表里点击 **grid1\_update\_onClick** 这个函数, 可以看到如下图所示的脚本设置界面:



跟打开添加页面一样, 修改页也是采用抽屉页的形式打开, 但是, 这里设置的页面路径不是固定的一个地址, 而是一个变量 **url**, 因为打开修改页面时, 不仅需要提供修改页面文件的

路径，还需要在 URL 路径里传入要修改的学员信息的 id，以便修改页面根据该 id 查询要修改的学员的信息，所以上面的这个函数设置了一个名为 `url` 的参数，该参数的值在数据表格的修改这一列里设置，具体设置方法如下：

在页面里选中数据表格组件 `grid1`，点击属性里的 **数据列设置**，打开如下所示对话框：



在左侧的列表里选中 **修改** 这一列，在右侧的基本信息设置里可以看到，在 **点击后执行的脚本** 输入框里有如下脚本：

```
grid1_update_onClick('update.tfp?id={id}')
```

该脚本里就是调用上面创建的那个函数，传入修改页面的路径，并且在 URL 路径里传入当前行的学员信息 id 值，这样就可以实现点击哪一行的修改按钮，就会打开修改页面显示这一行的学员信息，并可以对这个学员的信息进行修改。

#### 4.6.1.4 数据删除实现原理

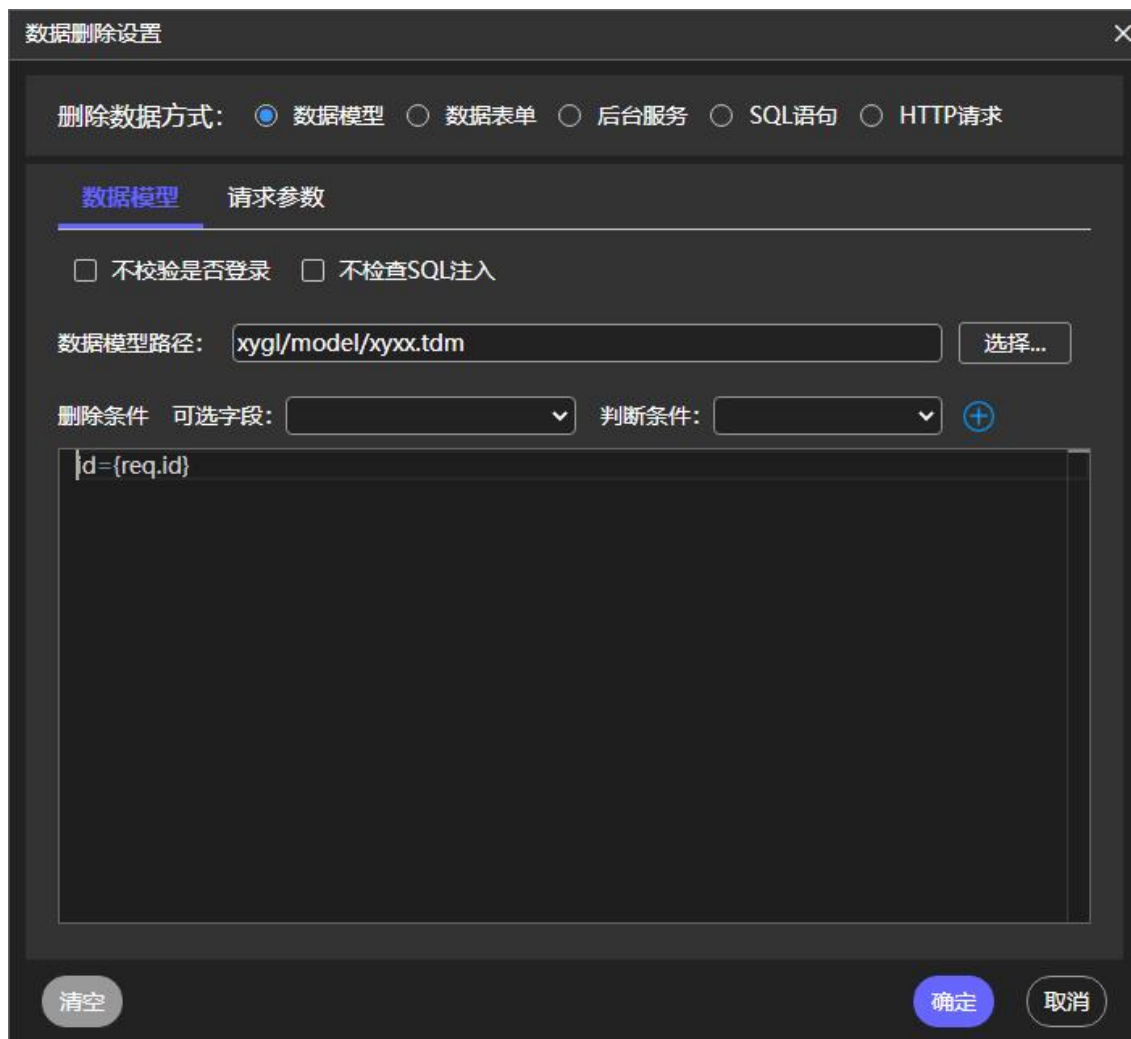
##### (1) 设置数据主键字段名

在数据表格组件 grid1 的属性里设置 **数据主键字段名**，以便后续在表格里点击删除按钮时，知道要给后台传哪个字段名，例如，学员管理示例项目里，主键字段名为 id，则设置界面如下：



## (2) 设置数据删除参数

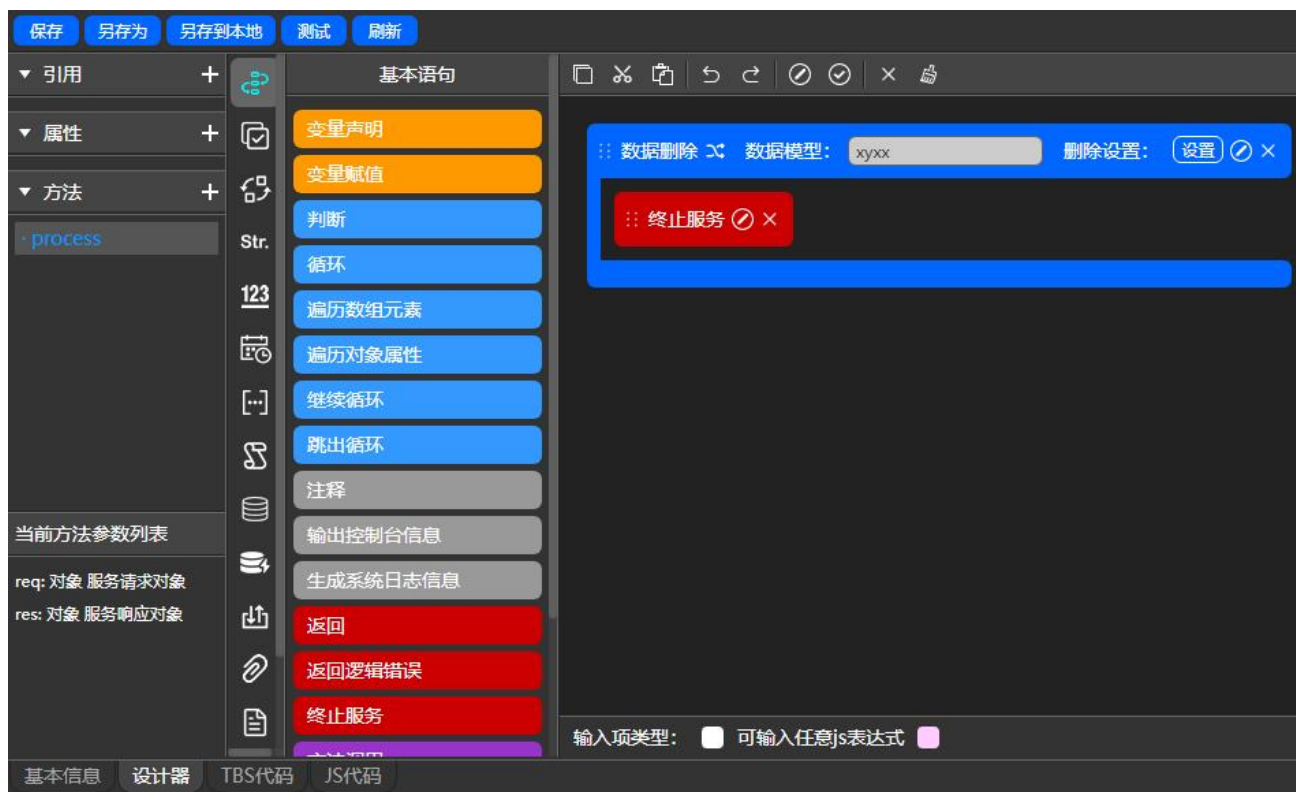
点击数据表格组件 grid1 属性设置里的“数据删除设置”，会打开如下所示对话框：



在此可以设置删除数据的 SQL 语句的 WHERE 条件，里面可以用 `{req.参数名}` 的形式插入客户端发送的请求参数。

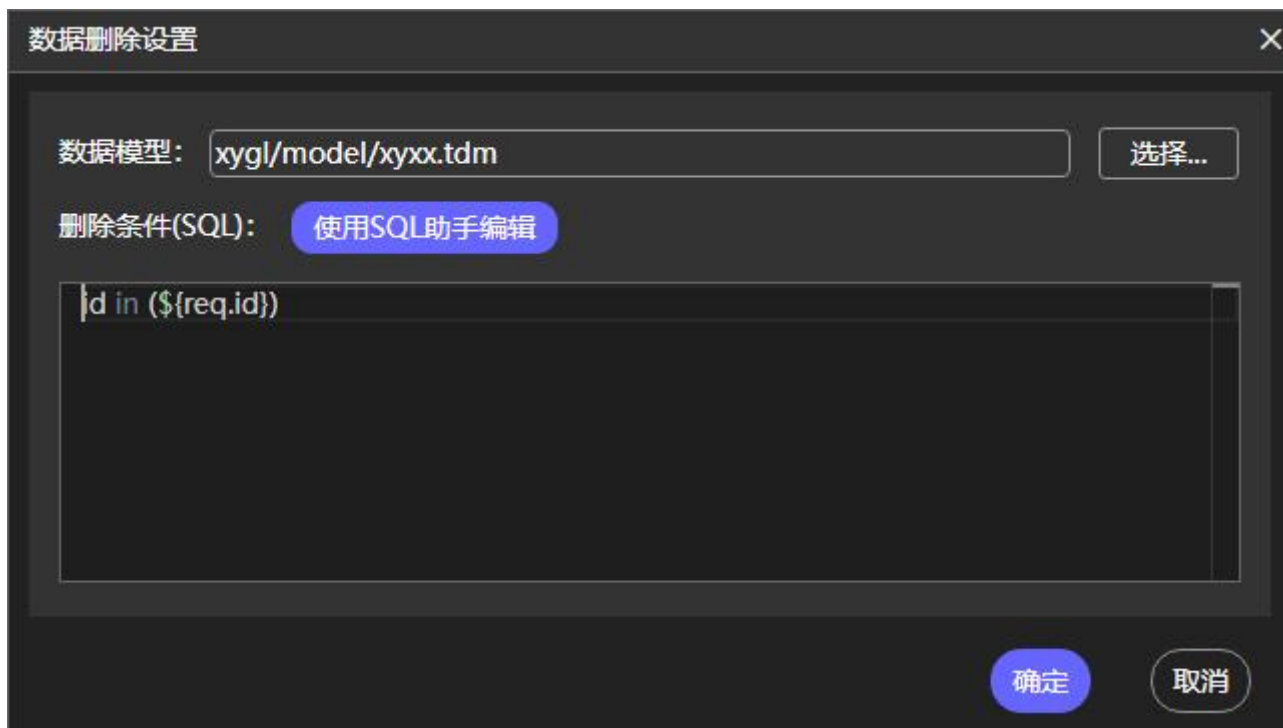
在此设置好数据删除有关的参数后，保存页面时，就会自动生成对应的删除数据的后台服务文件，展开学员管理项目的后台服务节点，在里面可以看到一个名为 `index_grid1_delete.tbs`

的后台服务文件，点击该文件，即可在后台服务设计器里打开，界面如下：



从上图可以看到，在该后台服务中，有一条 **数据删除** 语句，执行完后就终止后台服务。

点击数据删除语句的设置按钮，可以打开数据删除设置对话框，界面如下：



这里的设置实际就是在数据表格的数据删除设置里设置的参数，保存页面后自动生成的。

### (3) 设置删除按钮

在数据表格组件 grid1 的属性设置里点击 **数据列设置**，打开如下图所示的对话框，在左侧列表里选中 **删除** 这一列，在右侧的基本信息设置里可以看到，这一列的内容类型为 **按钮**，参数 **点击后执行的脚本** 设置成了如下所示的 JavaScript 脚本：

```
grid1.deleteRowData({id})
```

以上脚本就是调用数据表格组件的 `deleteRowData()` 方法来实现表格数据的删除。



## 4.6.2 数据添加页面前后端交互原理解析

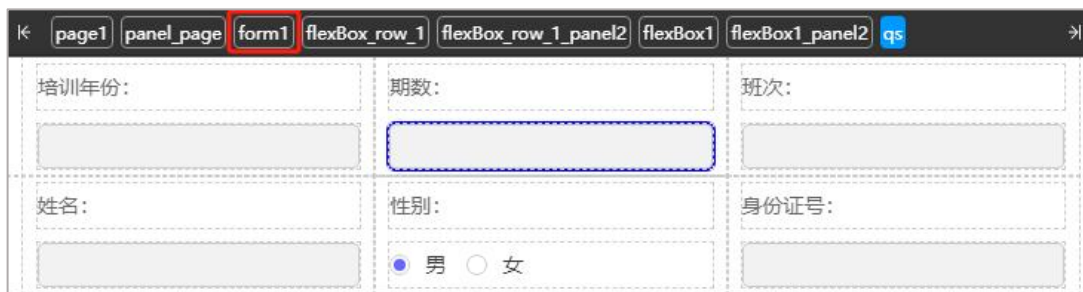
在 TaskBuilder 内打开学员管理项目前端文件里的 add.tfp 页面，下面以这个文件为例介绍一下数据添加页面如何实现前后端的数据交互。

如果要将页面里输入项的数据提交到后台，需要将这些输入项都放到表单组件中，然后通过表单组件实现数据的统一提交，具体原理如下：



### 4.6.2.1 设置表单组件的数据提交参数

在 add.tfp 页面内选中表单组件 form1，如果不知道怎么选择，可以在页面中随意选中一个输入项，然后在页面上部的组件路径导航栏里就能看到 form1，如下图所示：



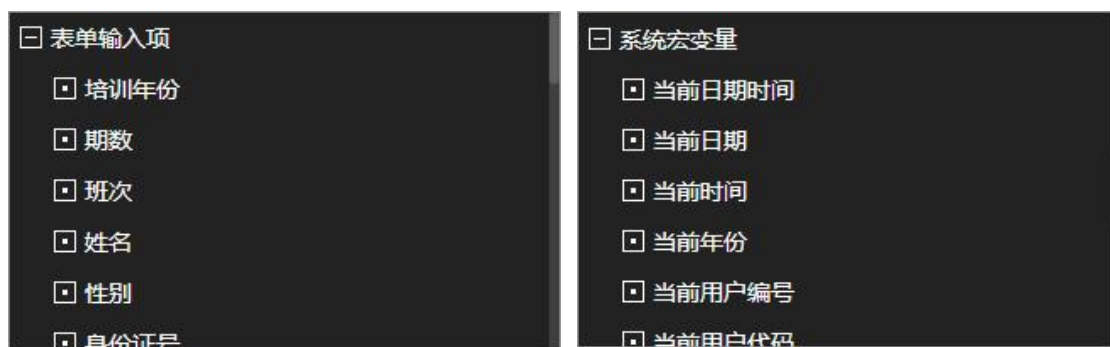
在这个导航栏里点击 form1 即可选中该表单组件，也可以点击左侧的 **组件大纲** 选项卡，在页面组件树形列表里选中 form1 组件。然后在属性设置里点击 **数据提交设置**，即可打开如下图所示的对话框：



在该窗口中，提交类型设置为 **添加**，表示该表单提交的数据是用来添加的，然后选择数



据模型，这里选的是在前面示例项目里创建的 xyxx.tdm，选择好数据模型后，就会在下面的字段列表里看到该模型中的所有字段，如果哪个字段要添加数据，就把该字段最左侧的复选框打上勾，然后设置字段值，点击字段值输入框，会显示一个弹出列表，在里面可以选择将当前表单里的输入项的值作为字段的值，也可以选择系统宏变量作为字段的值，界面如下所示：



如果选择的是表单输入项，则字段值会显示为：`req.输入项编号`，表示在后台服务里执行数据添加操作时，该字段的值会从前端请求对象 `req` 里获取，因为表单在提交数据时，会自动将这些输入项的值放到 `req` 对象中发送给后台服务。

如果选择的是系统宏变量，则会直接在后台服务中获取相应的值，不需要从客户端发送，例如当前日期时间的值：`new Date()`、当前用户编号的值：`req.session.userId` 等。

设置好这些参数后，点确定按钮即可完成设置，保存页面后就会自动生成对应的添加数据的后台服务文件。展开学员管理项目的后台服务节点，在里面可以看到名为 `add_form1_add.tbs` 的后台服务文件，点击该文件，即可在后台服务设计器中打开该后台服务，从设计界面可以看到，在该后台服务中有一个 **数据添加** 语句，然后在执行完添加后，会将新添加的学员信息的 `id` 值保存到服务响应对象 `res` 中，然后再终止后台服务，并将响应数据返回给客户端，详细的设置界面如下：



点击数据添加语句的设置按钮，可以看到如下图所示的设计界面：

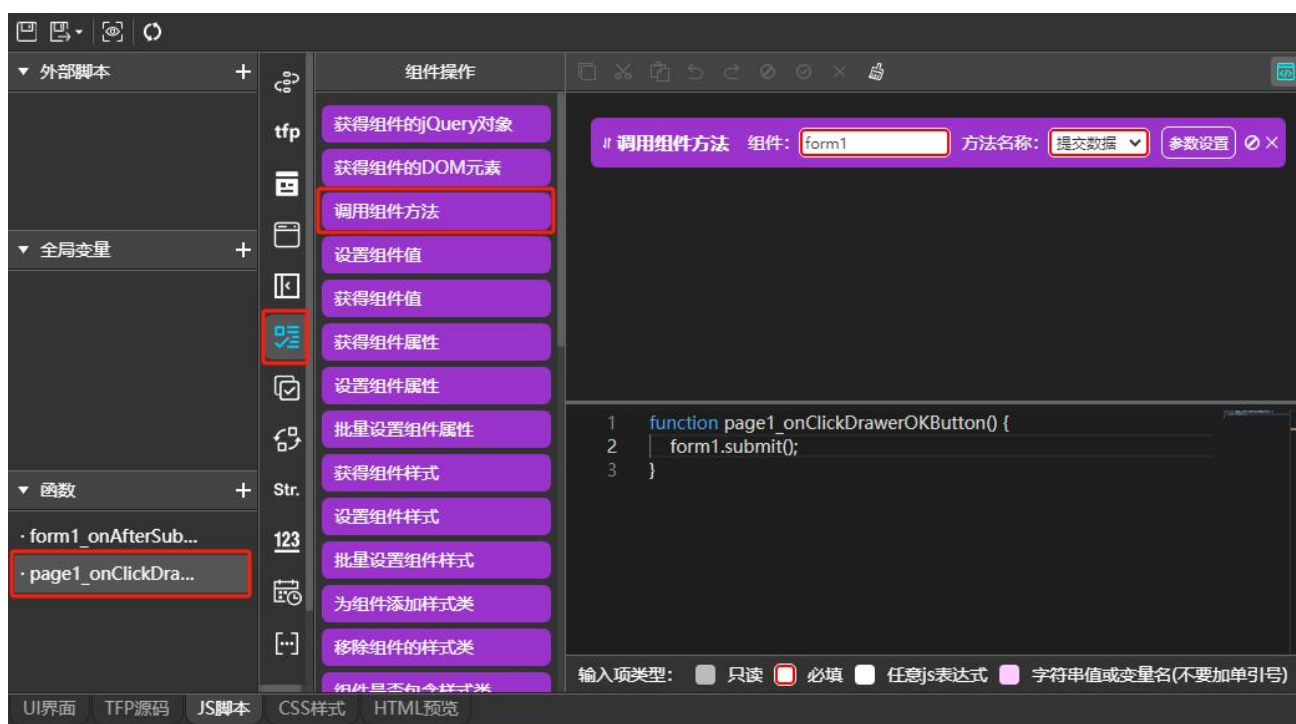


从上图可以看到，这些参数都是在 add.tfp 页面里的表单组件 form1 的数据提交设置里设置的，在用向导创建添加页面时会自动生成，在页面里修改后，再保存页面会重新生成。

#### 4.6.2.2 点击确定按钮时提交表单数据

表单数据的提交，一般是通过点击 **确定** 按钮来实现，用前端资源创建向导创建的添加页面默认是抽屉页形式，而抽屉页的确定和取消按钮是在系统顶层页面里统一提供的，抽屉页内并不需要添加确定按钮，所以，要实现表单数据的提交，需要通过设置页面的相关事件来实现，操作方法如下：

在前端页面设计器中，选择页面组件 page1，选择方法可以参考上面选择表单组件的方法。然后点击右侧的 **事件** 选项卡，双击 **点击抽屉页确定按钮时** 事件的输入框，会打开如下所示的脚本设置界面：



在该界面中可以看到，里面有一条 **调用组件方法** 的语句，组件设置为 form1，方法设置为提交数据，这样设置好后，当用户点击抽屉页顶部的 **确定** 按钮时，就会将表单里各个输入项的值全部放到 req 对象中发送给后台服务。

### 4.6.2.3 提交完成后更新列表数据

当表单提交完数据后，会触发表单的 **提交后** 事件，在该事件中可以设置相关脚本实现列表数据的刷新和关闭当前抽屉页，实现方法如下：

在 add.tfp 页面中选中 form1 表单组件，然后点击右侧的 **事件** 选项卡，就可以看到在 **提交后** 事件里已经设置了事件处理函数，双击该事件的输入框，可以看到如下所示的界面：



从上图可知，在该事件的处理函数内，提供了以下三条语句：

- (1) **获得当前抽屉页**：用该语句可以获得当前抽屉页对象，并保存到名为 drawer 的变量内；
- (2) **自定义语句**：通过 `drawer.opener` 可以获得打开抽屉页的上一个页面的 window 对象，然后调用这个页面里的数据表格组件 grid1 的重新加载数据方法 reloadData()实现列表数据刷新；
- (3) **关闭当前抽屉页**：最后关闭当前抽屉页。

这些语句都可以在左侧的语句库中找到，然后拖拽到右侧的业务逻辑编排区即可。

### 4.6.3 数据修改页面前后端交互原理解析

在 TaskBuilder 内打开学员管理项目前端文件里的 update.tfp 页面，下面以这个文件为例介

绍一下数据修改页面如何实现前后端的数据交互。

#### 4.6.3.1 表单数据查询设置

要在输入项中显示数据，需要将输入项放在表单组件里，并设置好表单组件的 **数据查询设置**，以学员管理项目的 update.tfp 页面为例，在页面中选中表单组件 form1，然后点击属性中的 **数据查询设置**，可以打开如下图所示的界面：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据模型** 请求参数 查询条件 分组和排序

☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径:

| <input type="checkbox"/>            | 字段名称  | 字段备注  | 字段格式(SQL, 双击可编辑) | 字段别名 |   |
|-------------------------------------|-------|-------|------------------|------|---|
| <input checked="" type="checkbox"/> | id    | 编号    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | pxnf  | 培训年份  |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | qs    | 期数    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | bc    | 班次    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | xm    | 姓名    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | xb    | 性别    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | sfzh  | 身份证号  |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | whcd  | 文化程度  |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | xz    | 乡镇    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | xzc   | 行政村   |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | cfakb | 是否贫困户 |                  |      | ⊖ |

在该对话框中，需要先选择好数据模型，然后选择要查询的字段。

点击 **请求参数** 选项卡，可以在这里设置需要客户端发送给后台服务的请求参数，对于修改页面来说，需要在此设置从列表页面通过 URL 传过来的 id 参数，设置界面如下：



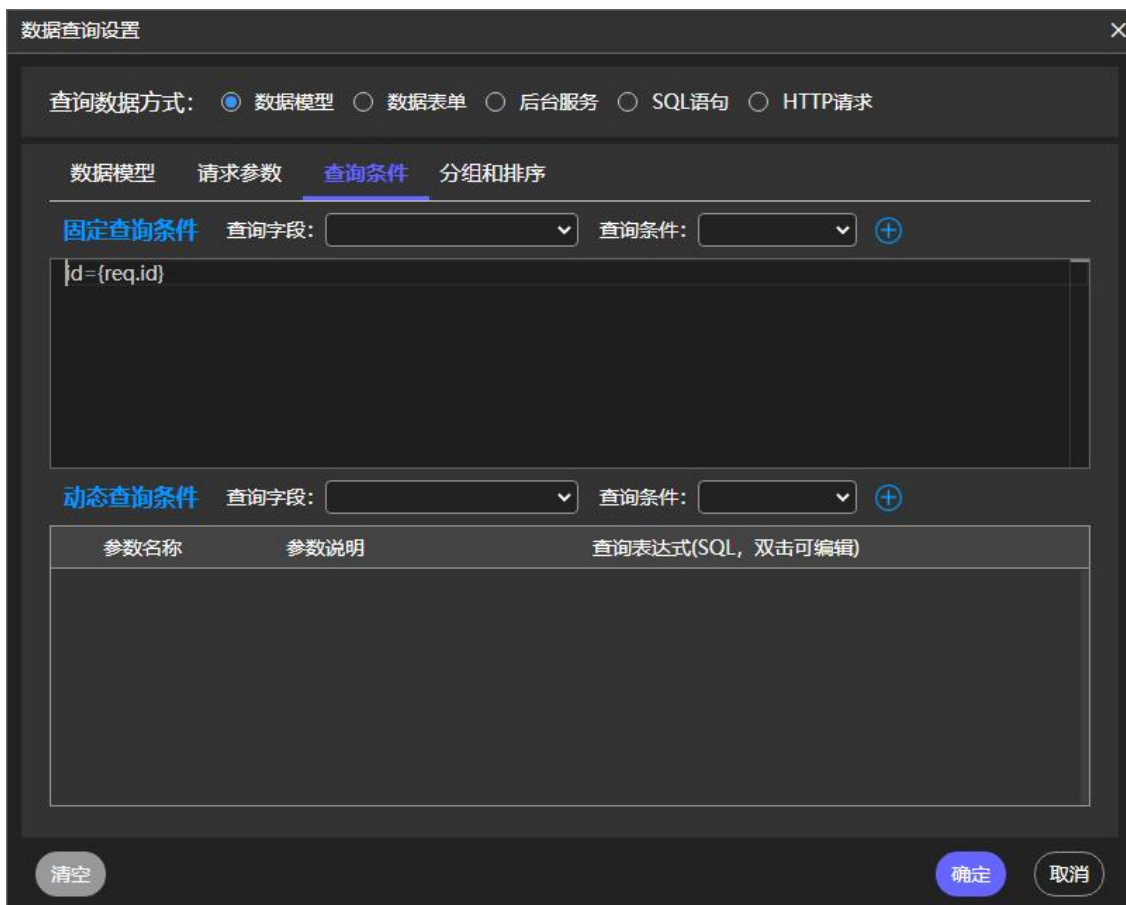
在上图的界面中，**参数名** 表示要传给后台服务的参数名称，**参数类型** 这里设置为 **URL 参数**，表示要从当前页面的 URL 路径里获取该参数的值，**参数值** 表示这个参数在 URL 里的名称，如果与要传给后台的参数名一致，则可以不用设置。大家可以回顾一下前面介绍数据列表页面的实现原理时，就讲解了如何在数据列表里打开修改页面，并通过 URL 路径给修改页面传递当前行数据的 id 值，正是因为列表页传了这个 id 参数，修改页才能通过该 id 值查询对应的学员信息，并在执行数据修改操作时，也才能知道具体要修改哪个学员的信息。

点击 **查询条件** 选项卡，可以查看和设置查询表单数据的 SQL 语句的查询条件，这个示例里设置的条件为：

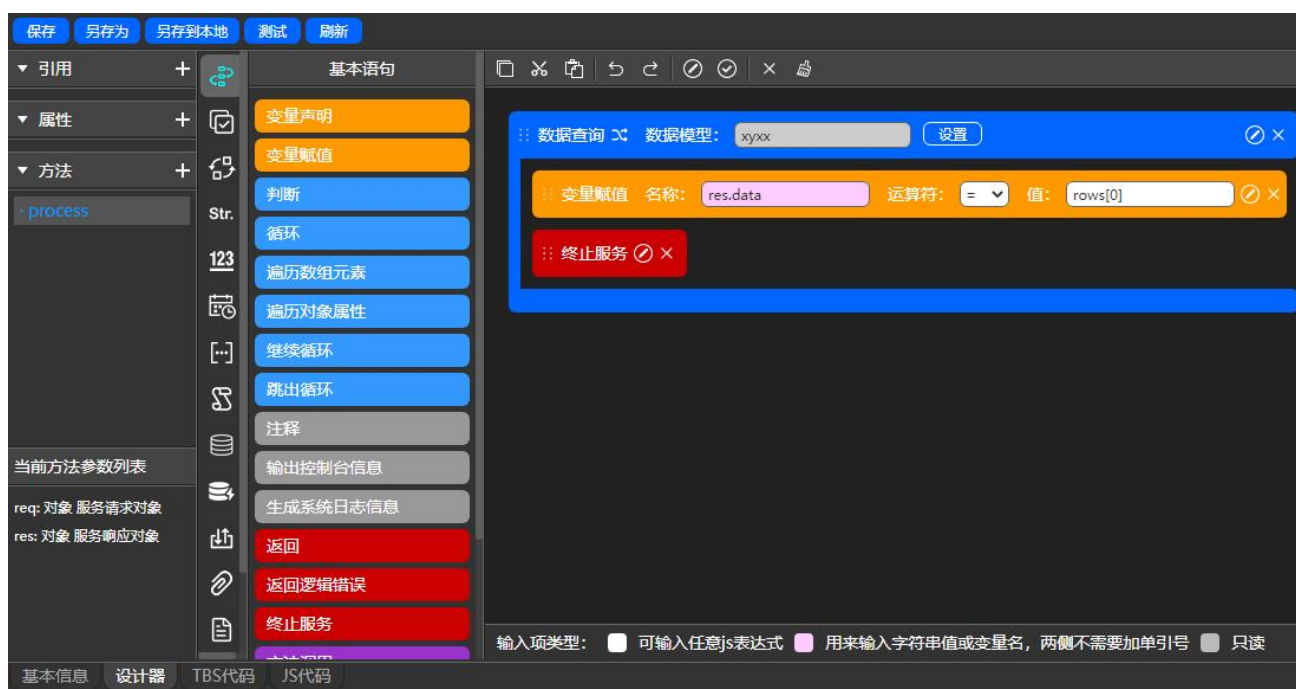
`id={req.id}`

上述查询条件表示在学员信息表中查询 id 字段的值等于前端传递给后台的 id 参数的值，设置界面如下图所示：





保存上述设置后，再保存页面，就会自动创建该数据查询设置对应的后台服务文件，学员管理示例项目的 update.tfp 页面生成的数据查询的后台服务名为 **update\_form1\_query.tbs**，在学员管理项目的后台服务里点击该文件，即可在后台服务设计器中打开，界面如下图所示：





从上图可知，在该后台服务中，最外层是一个数据查询语句，在查询完毕后，会将查询结果（一条数据记录）保存到服务响应对象 `res` 中，然后终止服务，并将 `res` 的内容返回给前端。

点击数据查询语句里的设置按钮，可以打开数据查询设置对话框对查询操作进行详细设置，设置内容和前面介绍数据列表页面的查询设置时的内容基本一致，区别主要是查询条件不一样，修改页面的查询，没有动态查询条件，只需要设置固定查询条件，就是根据列表页面传递的 `id` 值只查该 `id` 对应的学员信息，设置界面如下图所示：



#### 4.6.3.2 设置输入项数据绑定格式

设置好表单组件的数据查询参数后，接下来就需要设置每个输入项的 **数据绑定格式**，否则各个输入项并不知道要显示查询结果里的哪个字段的数据，设置方法如下：

在 `update.tfp` 页面中，选中某个输入项，例如编号为 `pxnf`（培训年份）的单行文本输入框，然后在其属性设置里，将 **数据绑定格式** 设置为：`{pxnf}`，表示在绑定表单查询到的数据时，要将查询结果里的 `pxnf` 这个字段的值绑定到这个输入项上，输入项的编号和字段名不一定要一样，只不过通过向导生成的页面里，输入项编号和要绑定到字段的名是一样的，实际上在 数  
北京任讯信息技术有限公司

**据绑定格式** 内，可以设置多个字段，并能跟静态内容拼接，例如：`{pxnf}年第{qs}`，如果查询结果里 pxnf 字段值为：2024，qs 字段的值为：2 期，则这个输入项的值为：**2024 年第 2 期**。

#### 4.6.3.3 请求后台服务查询并显示数据

在页面运行时，等页面加载完所有静态内容并解析渲染完毕后，表单组件会自动请求后台服务进行数据查询，并将查询结果绑定到各个输入项中。

#### 4.6.3.4 设置表单的数据提交参数

修改页面里表单组件的数据提交设置、确定按钮设置、提交后的事件设置与添加页面里的设置基本一致，区别是在修改页面里的表单组件在设置数据提交参数时，还需要设置请求参数和修改条件，具体设置方法如下：

在表单组件的数据提交设置对话框中，点击 **请求参数** 选项卡，可以看到如下界面：



数据提交设置

提交数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 **请求参数** 修改条件

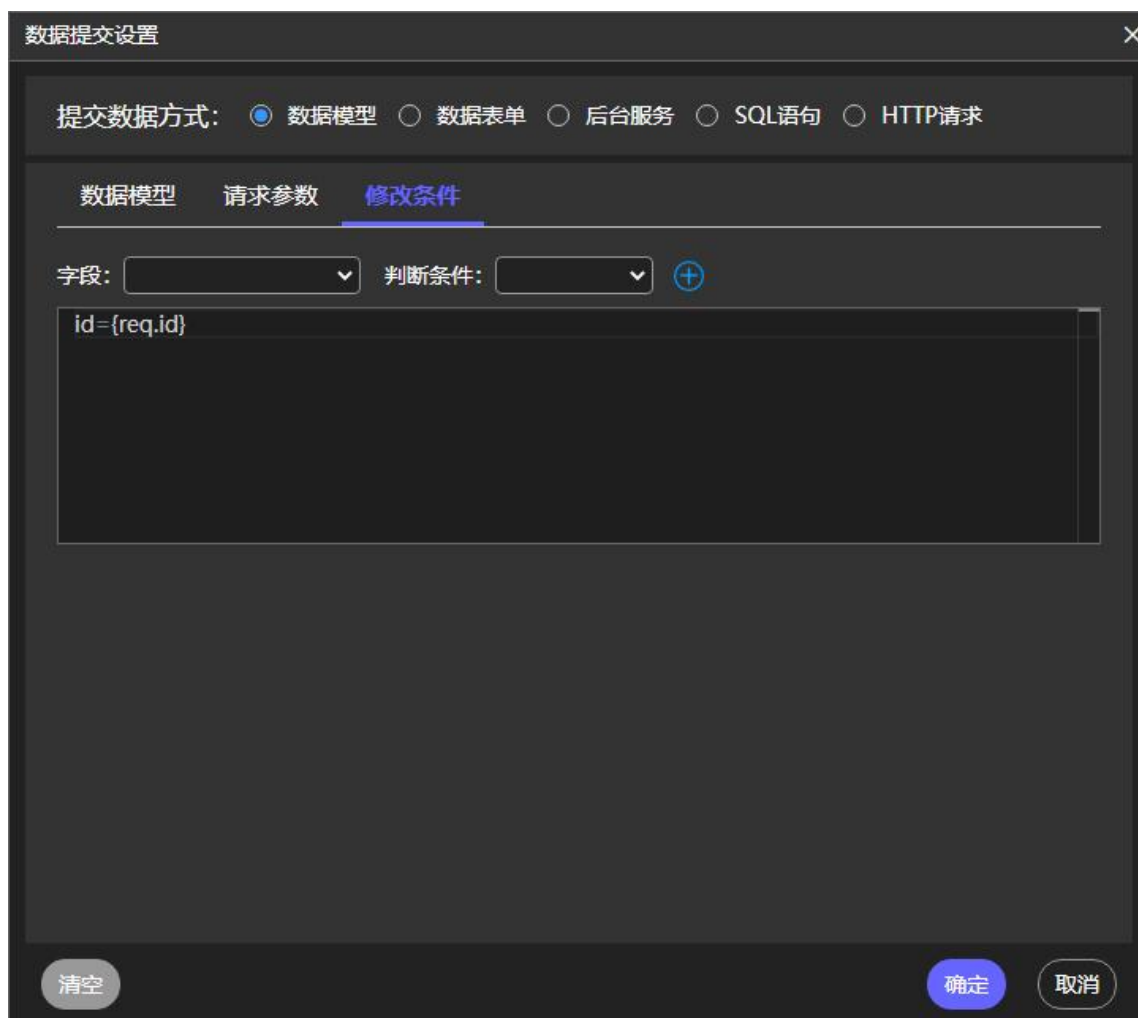
↓ 在此设置需要传递给后台的查询参数

| 参数名 | 参数类型  | 参数值 |
|-----|-------|-----|
| id  | URL参数 |     |

清空 确定 取消

可以在此设置需要客户端发送给后台服务的请求参数，对于数据修改操作来说，需要在此设置从列表页面通过 URL 传过来的 id 参数。

然后点击 **修改条件** 选项卡，可以看到如下界面：



数据提交设置

提交数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 **修改条件**

字段： 判断条件：

id={req.id}

在此可以设置修改数据的条件，这里设置成了：**id={req.id}**，表示只修改 id 字段的值等于前端传过来的 id 参数值的学员信息。

## 七、TaskBuilder 低代码开发项目实战

经过前面的学习，大家应该对 TaskBuilder 低代码开发有关的基本知识有了一个初步了解，接下来我们将带领大家实现一个较为复杂的信息系统，以便更深入地学习和掌握 TaskBuilder 提供的各项功能。

### 1、实战项目简介

本章准备带领大家开发一个简单的销售管理系统，希望通过该示例项目的实战操作，可以让大家掌握 TaskBuilder 低代码开发最基本的一些功能，然后有能力自主开发一些简单的管理信息系统。

该系统包括以下功能模块：

- **客户管理**：可以实现客户信息的增删改查，能关联查看客户有关的销售订单和收款信息。
- **产品管理**：可以实现产品类别和产品信息的增删改查。
- **销售订单**：可以实现销售订单的添加、审核和查询。
- **销售收款**：可以实现销售收款的登记、审核和查询。
- **销售统计**：可以实现销售数据的统计分析。

开发该系统的目的是为了讲解 TaskBuilder 的各项功能，与实际销售管理工作的要求有很大差距，请不要用于实际管理。

下面带领大家用 TaskBuilder 一步一步实现上面这些功能模块的开发。

### 2、创建项目

用 TaskBuilder 开发应用系统的第一步就是创建项目，项目可以是一个简单的功能模块，

也可以是很多功能模块的集合，具体怎么划分看各位的实际需要，我们一般会将相互关联比较紧密的一组功能模块放到一个独立的项目内，以便打包成独立的产品发布，例如人力资源管理、进销存、客户关系管理、项目管理等，或者为某个客户定制开发的项目，也会在 TaskBuilder 内创建独立的项目。

打开 TaskBuilder，点击左侧系统侧边栏里的“项目”图标，然后点击项目资源管理器右上角的蓝色加号图标，如下图：



即可打开新建项目对话框，如下图所示：

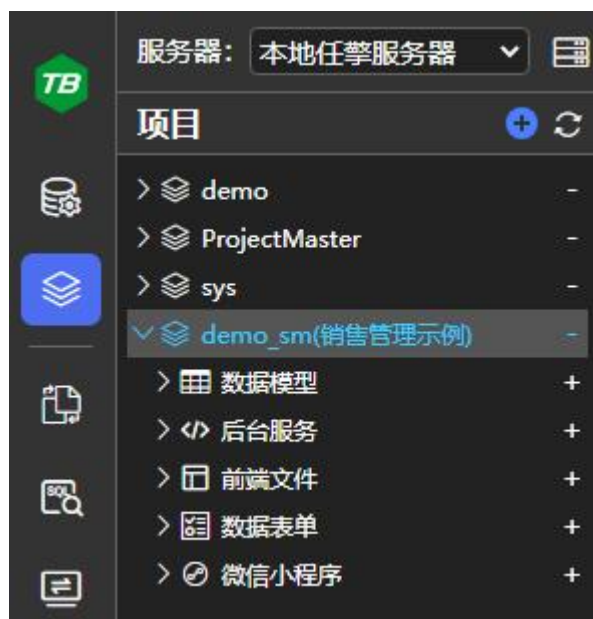


在该对话框中设置以下信息：

- **项目代码：** demo\_sm，sm 是 Sales Management 的缩写。
- **项目说明：** 销售管理示例
- **后台语言：** JavaScript

然后点确定，即可完成项目的创建，然后在 TaskBuilder 的项目资源管理器内就可以看到

该项目，点击该项目节点可以展开该项目，看到如下所示的界面：



项目创建好之后，就可以在里面开发各个功能模块了。

## 3、客户管理模块开发

### 3.1 本节知识点

通过本节的学习，可以掌握以下知识点：

- (1) 数据模型设计常用知识；
- (2) 使用前端资源创建向导创建数据查询、添加和修改页面详解；
- (3) 数据表格组件常用设置；
- (4) 前后端调试信息的查看；
- (5) 抽屉页的使用；
- (6) 表单组件的使用；
- (7) 数据字典的使用；
- (8) 用户选择组件的使用；

- (9) 地区选择组件的使用;
- (10) 静态列表组件的使用;
- (11) 设置应用分类和应用;
- (12) 应用访问权限设置;
- (13) 应用操作权限设置;

## 3.2 创建客户信息数据模型

开发一个应用模块的第一步是设计数据模型, 因为后续在开发前后端功能时都需要使用到数据模型, 现在带领大家创建和设计客户信息的数据模型, 具体操作如下:

### 3.2.1 使用向导创建数据模型

点击销售管理示例项目内数据模型节点右边的加号, 打开如下图所示的“添加数据模型资源”对话框:



在该对话框中选中“新建表”, 然后点下一步, 会打开“新建数据模型”对话框, 在该对话框中设置以下信息:

- **模型名称:** demo\_sm\_customer
- **模型备注:** 客户信息

为了方便大家快速创建数据模型的字段, 在该对话框下部的列表中提供了一些常用的字段,



这些字段已经设置好了相关信息，包括字段名、数据类型、长度、备注、是否自增等，如果我们要创建的数据模型需要用到这些字段，则在列表中点击这些字段最左侧的复选框即可选中他们，然后点确定按钮后，即可创建对应的数据模型模板，里面就自动创建好了选中的这些字段，不需要开发者再手动添加和设置，只需要再添加其他没有的字段即可。对于这里要创建的客户信息来说，我们可以选中以下字段：

id、code、name、creator\_id、create\_time

新建数据模型

模型路径：

/app/demo\_sm/model

模型名称：

demo\_sm\_customer

只能是字母、数字和下划线，且以字母开头

模型备注：

客户信息

数据源：

任擎默认数据库

创建常用字段：

| <input type="checkbox"/>            | 字段名          | 数据类型     | 长度 | 字段备注  | 其他            |
|-------------------------------------|--------------|----------|----|-------|---------------|
| <input checked="" type="checkbox"/> | id           | INT      | 11 | 编号    | 主键，自动增长       |
| <input checked="" type="checkbox"/> | code         | VARCHAR  | 20 | 编码    |               |
| <input checked="" type="checkbox"/> | name         | VARCHAR  | 20 | 名称    |               |
| <input type="checkbox"/>            | parent_id    | INT      | 11 | 父级编号  | 默认为0          |
| <input checked="" type="checkbox"/> | creator_id   | INT      | 11 | 创建人编号 |               |
| <input checked="" type="checkbox"/> | create_time  | DATETIME |    | 创建时间  |               |
| <input type="checkbox"/>            | auditor_id   | INT      | 11 | 审核人编号 |               |
| <input type="checkbox"/>            | audit_time   | DATETIME |    | 审核时间  |               |
| <input type="checkbox"/>            | audit_result | VARCHAR  | 20 | 审核结果  | Y: 同意, N: 不同意 |

确定

取消

设置好以上信息后，点确定按钮，即可在 TaskBuilder 的数据模型设计器中打开该模型，界面如下所示：



从上图可以看到，我们刚才选的那些常用字段已经自动创建好，我们只需要再添加其他客户有关的字段即可。

### 3.2.2 数据模型设计器使用简介

点击上图所示 **数据模型设计器** 上部工具栏上的 **添加字段** 按钮，即可在字段列表底部添加新的一行，可以在这里设置新字段的以下信息：

- **字段名**：用来设置字段的名称，只能是字母、数字或下划线，且必须以字母开头，且不能使用 SQL 的关键字，例如：select、update、delete、create、drop、alter、int、varchar、int、datetime、date、now、year、month 等（不限于这些，具体可以参考相关数据库的文档）。实际上汉字也可以，但不推荐。
- **字段类型**：用来设置字段的数据类型，常用类型包括：
  - (1) **字符串**：用来存储多个字符信息，需要设置合理的最大长度，以便合理分配存储空间。
  - (2) **整数**：用来存储整数数字。

- (3) **小数**：用来存储小数数字。
  - (4) **金额**：用来存储金额数据，默认精确到小数点后两位。
  - (5) **数值**：用来存储对精度有要求的小数，例如税率、汇率等，可以设置精度（保留指定小数位数），超过位数时自动舍入。
  - (6) **日期时间**：用来存储日期（年月日）+时间（时分秒）数据，精确到秒。
  - (7) **字符**：用来存储单个字符，例如：Y、N。
  - (8) **文本**：用来存储长文本信息，例如文本文件的内容，长度可能超过 5000 的文本数据都建议使用文本类型，不建议使用字符串类型，因为字符串类型有最大长度限制。
  - (9) **其他数据类型**：不常用的类型在此不挨个介绍，需要详细了解的，可以自己搜索查找您所使用的数据库关于数据类型的介绍文档。
- **长度**：用来设置当前字段可以存储数据的最大长度，主要字符串和数值这两种类型需要设置长度，其他数据类型可以不用设置。
  - **精度**：用来设置金额和数值类型数据的精度（保留小数点位数），默认为 2 位，可以根据需要修改。
  - **主键**：用来设置当前字段是否为当前数据模型的主键，主键是一种约束规则，可以是单独的一个字段，也可以由多个字段组合（将多个字段设置为主键），主键字段组成的值能唯一地标识表中的每一行，通过它可强制约束表的实体完整性。主键可以方便后续进行查询、修改和删除等操作，也可以用于与其他表进行外键关联。
  - **非空**：用来设置字段值是否允许为空，如果将字段设置为主键或自动增长，则会自动设置为不允许为空。
  - **自增**：用来设置字段值是否是自动增长，如果设置为自动增长，则字段类型会自动设置为整数，且不允许为空。

- **唯一**：用来设置字段值是否允许重复，如果设置为唯一，则每条记录的这个字段的值必须是唯一的，不能重复。
- **默认值**：用来设置字段的默认值，如添加数据时，没有为该字段设置值，则会采用此处设置的默认值。默认值如果是字符串，需要在值两边加单引号，如果是数值，则不需要加单引号。不支持使用任何 SQL 函数和表达式，只能是具体的值。
- **备注**：用来设置字段的备注，建议用汉字。

点击每个字段信息所在行最右侧的上移、下移按钮，可以调整字段位置；点击删除按钮，可以删除该字段；也可以点击字段信息前面的选择框，选中多个字段后，再点击工具栏上的 **删除字段** 按钮，批量删除多个字段。

### 3.2.3 数据模型字段设置注意事项

在设置数据模型的字段信息时，请参照以下原则设置：

(1) 建议每个表都至少将一个字段设置为主键，推荐使用 id 作为主键字段名，且设置为自增长，即每添加一条记录，id 值就自动加 1，后续使用 TaskBuilder 向导创建相关前后端文件时，都默认数据模型的主键字段名为 id；

(2) 如果字段的类型为字符串，请设置适当的长度，不要过长，浪费存储空间，也不能太短，造成实际业务数据无法保存；

(3) 如果只需要保存 4 位年份，字段类型请使用整数或字符串，不要使用“日期时间”这种类型，因为这种类型的数据，在添加记录时，即使给这个字段设置的值只有年，但实际上保存到数据库里时，会自动生成年月日和小时分钟秒的完整日期时间值，所以后续要跟 4 位的年份值进行比对时就会造成与实际不符。同样，如果只需要保存年月日的值，也不建议用“日期时间”类型，推荐用 10 位字符串；

(4) 对于值不能为空的字段，除了在数据模型里要将该字段设置为非空，在添加和修改

该数据模型的后台服务里，如果这个字段的值是由前端页面传递过来的，则要在该后台服务的基本信息里将该字段对应的前端请求参数设置为不能为空，并设置参数名称，这样在前端页面请求该服务时，如果这个参数没有传值，就会自动返回该参数不能为空的错误提示，否则，会显示数据库软件返回的错误提示，基本都是英文，而且提示不明确，对于用户来说非常不友好；

(5) 如果字段值不能重复，光在数据模型里设置成“唯一”是不够的，在添加和修改该数据模型的后台服务里，需要先通过数据查询语句判断是否已存在重复的记录，如果没有重复再执行实际的添加或修改操作，否则，数据库软件会报错，也都是英文，对用户不友好；

(6) 请为每个字段设置好备注，且建议用汉字，因为在后续使用 TaskBuilder 进行应用开发时，会经常用到字段的备注信息，例如使用前端页面向导创建数据查询页面时，如果为字段设置里备注，则用字段的备注作为数据表格里对应列的标题，否则会用字段名（可能是英文）作为列名；在创建添加和修改页面时，也会用字段备注作为输入项的文本标签了的文字。所以，为了方便后续开发，建议为每个字段都设置好备注。

### 3.2.4 设置客户信息数据模型的字段

客户信息除了上面选择的那几个常用字段，还需要在数据模型设计器里添加以下字段：

| 字段名         | 数据类型 | 长度 | 字段备注  | 非空 | 唯一 | 说明         |
|-------------|------|----|-------|----|----|------------|
| status      | 字符串  | 2  | 客户状态  | 是  |    | 关联客户状态数据字典 |
| salesman_id | 整数   | 11 | 销售员编号 | 是  |    | 关联系统用户信息   |
| contacts    | 字符串  | 10 | 联系人   | 是  |    |            |
| telephone   | 字符串  | 20 | 联系电话  | 是  |    |            |
| scale       | 字符串  | 2  | 客户规模  | 是  |    | 关联客户规模数据字典 |
| grade       | 字符串  | 2  | 客户等级  | 是  |    | 关联客户等级数据字典 |

|             |      |     |       |   |  |            |
|-------------|------|-----|-------|---|--|------------|
| source      | 字符串  | 2   | 客户来源  | 是 |  | 关联客户来源数据字典 |
| trade       | 字符串  | 2   | 所属行业  | 是 |  | 关联所属行业数据字典 |
| site        | 字符串  | 200 | 客户网址  |   |  |            |
| province    | 字符串  | 10  | 所在省份  | 是 |  | 关联系统省市区编码  |
| city        | 字符串  | 10  | 所在城市  | 是 |  |            |
| address     | 字符串  | 300 | 详细地址  |   |  |            |
| comment     | 字符串  | 500 | 客户备注  |   |  |            |
| deletor_id  | 整数   | 11  | 删除人编号 |   |  |            |
| delete_time | 日期时间 |     | 删除时间  |   |  |            |

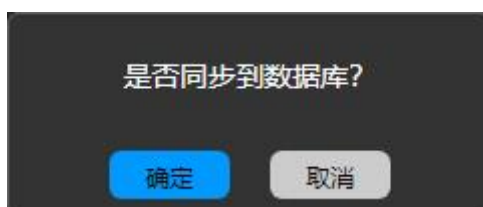
除了上表中的字段，原来用向导自动创建的字段还需要进行以下设置：

- (1) 将 code、name、creator\_id、create\_time 这几个字段设置为非空；
- (2) 将 code、name 这两个字段设置为唯一（不能重复）；

设置完之后的客户信息数据模型设计界面如下图所示：

| 字段名         | 字段类型 | 长度  | 精度 | 主键                                  | 非空                                  | 自增                                  | 唯一                                  | 默认值 | 备注    |
|-------------|------|-----|----|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-----|-------|
| id          | 整数   | 11  |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |     | 编号    |
| code        | 字符串  | 20  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |     | 编码    |
| name        | 字符串  | 20  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/> |     | 名称    |
| creator_id  | 整数   | 11  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 创建人编号 |
| create_time | 日期时间 |     |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 创建时间  |
| status      | 字符串  | 2   |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 状态    |
| salesman_id | 整数   | 11  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 销售员编号 |
| contacts    | 字符串  | 10  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 联系人   |
| telephone   | 字符串  | 20  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 联系电话  |
| scale       | 字符串  | 2   |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 客户规模  |
| grade       | 字符串  | 2   |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 客户等级  |
| source      | 字符串  | 2   |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 客户来源  |
| trade       | 字符串  | 2   |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 所属行业  |
| site        | 字符串  | 200 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 客户网址  |
| province    | 字符串  | 10  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 所在省份  |
| city        | 字符串  | 10  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 所在城市  |
| address     | 字符串  | 300 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 详细地址  |
| comment     | 字符串  | 200 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 备注    |
| deleter_id  | 整数   | 11  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 删除人编号 |
| delete_time | 日期时间 |     |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            |     | 删除时间  |

都设置之后，点击上部工具栏里的“保存”按钮，会弹出以下提示：



如果点“确定”，则该数据模型的信息除了以文件形式保存到任擎服务器上，而且还会连接数据库，创建对应的表，如果点取消，则只会保存为文件，不会连数据库建表，这样做主要是为了先存一个草稿，过后还可以打开修改，都设置好之后，再同步到数据库。这里我们点“确定”，完成客户信息数据模型的设计。

### 3.2.5 数据模型设计器工具栏介绍

当我们保存完上述客户信息数据模型后，会发现数据模型设计器顶部的工具栏上多了以下



按钮：



他们的功能如下：

- (1) **另存为**：点击后可以将当前数据模型另存为其他名称的数据模型文件。
- (2) **从数据库同步**：如果在数据库里修改了当前数据模型对应表的结构或设置，则可以点击该按钮将数据库里的改动信息同步到当前数据模型文件中；
- (3) **同步到数据库**：如果在创建完数据模型，后面又在数据模型设计器中修改了该模型的相关内容，则可以点该按钮将改动同步到数据库；
- (4) **查询数据**：点击后可以打开数据查询窗口，会自动生成一条查询该数据模型数据的 SQL 查询语句，然后执行并显示查询结果，也可以修改该 SQL 语句，查询自己想要的数据。

### 3.2.6 数据模型设计器其他功能

另外，在数据模型设计器底部，还通过选项卡的形式提供了几项功能：



#### 3.2.6.1 基本信息

点击数据模型设计器底部的“基本信息”选项卡，可以查看和设置数据模型的基本信息，界面如下图所示：

|         |                      |
|---------|----------------------|
| 模型文件名:  | demo_sm_customer.tdm |
| 数据源ID:  | default              |
| 数据库类型:  | mysql                |
| 数据对象类型: | 表                    |
| 同步状态:   | 已同步                  |
| 模型备注:   | 客户信息                 |

在此，可以查看当前数据模型的文件名、数据源 ID、数据库类型、数据对象类型、修改内容是否已同步、模型备注等，其中模型备注可以修改，建议为每个模型设置备注，这样在后续选择数据模型时，会在模型列表里显示备注，就可以很清楚地了解每个模型具体是做什么的。

### 3.2.6.2 TDM 代码

点击数据模型设计器底部的“TDM 代码”选项卡，可以查看当前数据模型的 tdm 源码，目前不允许修改，只能查看，内容为 JSON 格式，里面定义了当前数据模型的基本信息和所有字段的信息，如下图所示：

```
1  {
2    "table": "demo_sm_customer",
3    "type": "table",
4    "version": 1,
5    "fields": {
6      "id": {
7        "dataType": "INT",
8        "type": "int",
9        "primaryKey": true,
10       "autoIncrement": true,
11       "notNull": true,
12       "uniqueIndex": false,
13       "length": 11,
14       "default": "",
15       "comment": "编号",
16       "version": 1,
17       "status": 0,
```

### 3.2.6.3 创建模型 DDL

点击数据模型设计器底部的“创建模型 DDL”选项卡，可以查看创建当前数据模型对应的数据表的 SQL 语句，界面如下图所示：

```

1 CREATE TABLE `demo_sm_customer` (
2   `id` INT NOT NULL AUTO_INCREMENT COMMENT '编号',
3   `code` VARCHAR(20) NOT NULL COMMENT '编码',
4   `name` VARCHAR(20) NOT NULL COMMENT '名称',
5   `creator_id` INT NOT NULL COMMENT '创建人编号',
6   `create_time` DATETIME NOT NULL COMMENT '创建时间',
7   `deletor_id` INT COMMENT '删除人编号',
8   `delete_time` DATETIME COMMENT '删除时间',
9   `status` TINYINT NOT NULL DEFAULT 0 COMMENT '状态',
10  `comment` VARCHAR(200) COMMENT '备注',
11  `scale` VARCHAR(2) NOT NULL COMMENT '客户规模',
12  `trade` VARCHAR(2) NOT NULL COMMENT '所属行业',
13  `grade` VARCHAR(2) NOT NULL COMMENT '客户等级',
14  `source` VARCHAR(2) NOT NULL COMMENT '客户来源',
15  `contacts` VARCHAR(10) NOT NULL COMMENT '联系人',
16  `telephone` VARCHAR(20) NOT NULL COMMENT '联系电话',
17  `site` VARCHAR(200) COMMENT '客户网址',
18  `province` VARCHAR(10) NOT NULL COMMENT '所在省份',
19  `city` VARCHAR(10) NOT NULL COMMENT '所在城市',
20  `address` VARCHAR(300) COMMENT '详细地址',
21  `salesman_id` INT NOT NULL COMMENT '所属销售人员id',
22  PRIMARY KEY(`id`),
23  UNIQUE KEY `un_demo_sm_customer_code`(`code`),
24  UNIQUE KEY `un_demo_sm_customer_name`(`name`)
25 ) ENGINE=INNODB CHARSET=UTF8MB4 COMMENT='客户信息';
26

```

将该语句复制到其他数据库中执行，可以实现跨库的表结构复制。

### 3.2.6.4 变更模型 DDL

点击数据模型设计器底部的“变更模型 DDL”选项卡，可以查看修改当前数据模型后对应的变更数据库表结构的 SQL 语句，例如修改了客户信息数据模型里的 status 字段的数据类型和 comment 字段的长度，则对应的修改表结构的 SQL 语句如下图所示：

```

1 ALTER TABLE `demo_sm_customer`
2   CHANGE COLUMN `status` `status` INT NOT NULL DEFAULT 0 COMMENT '状态' AFTER `delete_time`,
3   CHANGE COLUMN `comment` `comment` VARCHAR(300) COMMENT '备注' AFTER `status`,
4   COMMENT='客户信息';

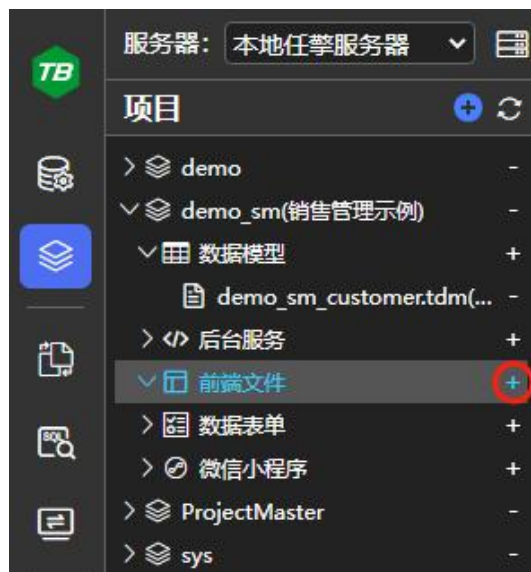
```

将该语句复制到其他数据库中执行，可以实现跨库的表结构修改的同步。

### 3.3 创建客户信息文件夹

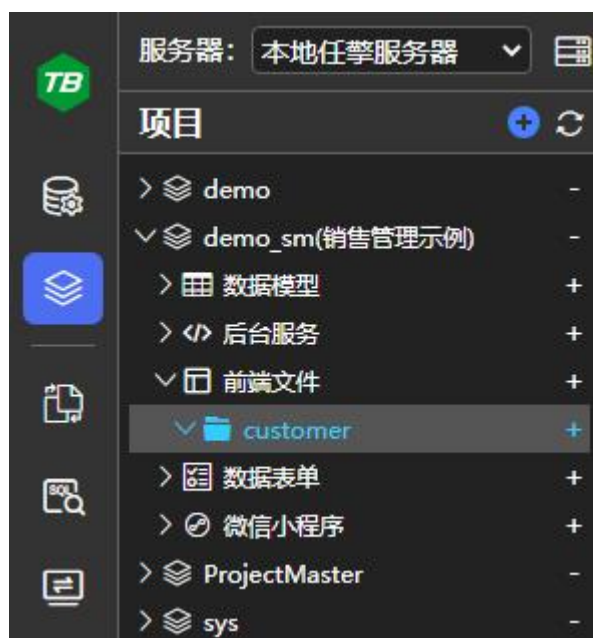
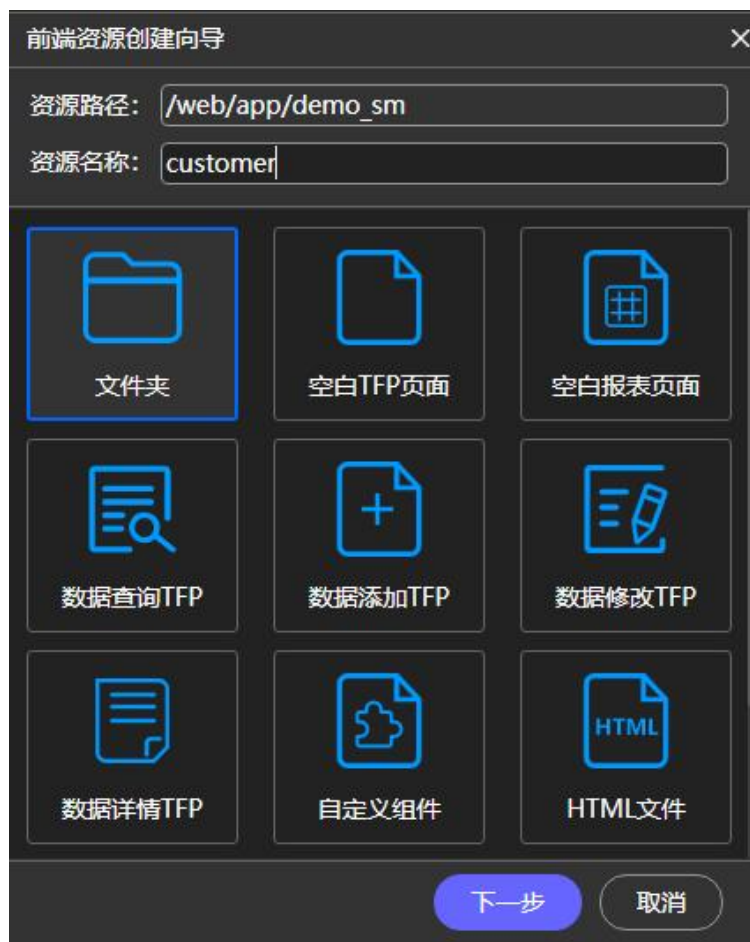
数据模型创建好之后，我们就可以进行前后端功能的开发了。首先，我们需要创建好客户信息文件夹，以便专门存放与客户信息管理有关的前端文件，操作步骤如下：

点击销售管理示例项目“前端文件”右侧的加号按钮：



会打开如下图所示的“前端资源创建向导”对话框，在该对话框中，选中“文件夹”，在资源名称输入框中输入：customer，然后点“下一步”按钮，就可以在销售管理示例项目的前端创建一个名为“customer”的文件夹，以便存放所有客户管理有关的前端文件。

不需要在后台服务里手动创建客户信息的文件夹，在用向导创建前端页面时，如果选择了自动创建页面所需的后台服务，向导会根据前端文件的路径，自动创建后台服务对应的文件夹。



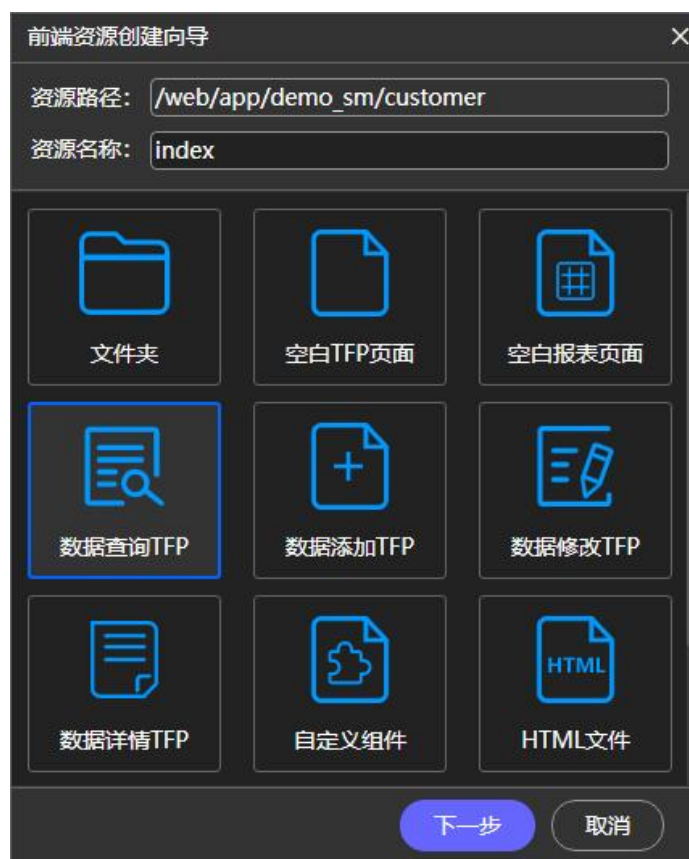
如果一个项目内有多个功能模块，建议为每个模块都创建一个单独的文件夹，然后把这个模块的相关文件都存放在这个文件夹里，不要把一个项目的所有文件都放到一个文件夹里，那样不便于查看和维护。



## 3.4 创建客户信息列表页面

### 3.4.1 选择页面类型

点击上面创建的 customer 文件夹右侧的加号，打开“前端资源创建向导”对话框，选中“数据查询 TFP”，资源名称会自动设置为 index，这里我们不用改。



点“下一步”按钮，会弹出下图所示的“创建数据查询 TFP”的对话框：

### 3.4.2 设置基本信息

在该对话框中，第一步需要设置页面的基本信息，包括以下选项：

- **保存路径：**根据项目代码、所属文件夹和文件名自动生成，在此不能修改；
- **数据模型：**要查询数据的表或视图对应的数据模型文件的路径，通过点击右侧的“选择...”按钮选择；

- **页面标题**：页面打开后在系统标签栏内显示的标题，不能为空；
- **页面类型**：支持普通页面和对话框两种类型，默认为普通页面；
- **窗口大小**：用来设置对话框页面的宽度和高度，单位为像素（px），如果是普通页面，则不需要设置；
- **终端类型**：用来设置访问当前页面的终端类型，目前支持电脑端、手机 H5、任讯 APP、阿里钉钉和企业微信这几种类型，默认为电脑端；
- **背景颜色**：在此可以设置页面的背景颜色是深色还是浅色，以便决定页面内各个组件的样式，默认为浅色；
- **定位方式**：在此可以设置页面里新建组件默认的定位方式，支持浮动定位和绝对定位两种方式，默认为浮动定位；

以上页面基本信息的各个选项的具体含义可以查看本文的第 3.3.2 小节“前端页面基本信息设置”里的详细介绍，这里就不重复讲解了。

The screenshot shows a dialog box titled "创建数据查询TFP" (Create Data Query TFP) with a close button (X) in the top right corner. The main content area is titled "第一步：设置基本信息" (Step 1: Set Basic Information). The form includes the following fields and options:

- 保存路径** (Save Path): /web/app/demo\_sm/index.tfp
- 数据模型** (Data Model): [Empty field] with a "选择..." (Select...) button.
- 页面标题** (Page Title): [Empty field]
- 页面类型** (Page Type): 普通页面 (Normal Page) [Dropdown]
- 窗口大小** (Window Size): 宽 (Width) [Input] 高 (Height) [Input]
- 终端类型** (Terminal Type): 电脑端 (Desktop) [Dropdown]
- 背景颜色** (Background Color): 浅色 (Light) [Dropdown]
- 定位方式** (Positioning Method): 浮动定位 (Floating Positioning) [Dropdown]
- 查询设置** (Query Settings):
  - ☒ 启用翻页 (Enable Paging)
  - 每页最多显示 (Max items per page): 20 [Dropdown]
  - 条记录 (Records) [Input] 最多查询 (Max queries) [Input] 条记录 (Records) [Input]
  - ☐ 去掉重复项 (Remove duplicates)
- 其他设置** (Other Settings):
  - ☒ 显示添加按钮 (Show Add button)
  - 添加页面文件名 (Add page filename): add.tfp [Input]
  - 添加页面类型 (Add page type): 抽屉页 (Drawer Page) [Dropdown]
  - ☒ 显示修改按钮 (Show Modify button)
  - 修改页面文件名 (Modify page filename): update.tfp [Input]
  - 修改页面类型 (Modify page type): 抽屉页 (Drawer Page) [Dropdown]
  - ☒ 显示删除按钮 (Show Delete button)
  - ☐ 查询条件显示在页面左侧 (仅支持PC端) (Query conditions displayed on the left side of the page (PC only))
- 数据查询服务** (Data Query Service): ☒ 自动创建 (Auto Create) ☐ 使用已有服务 (Use Existing Service) 响应参数名 (Response parameter name): data [Input] 选择... [Button]
- 数据删除服务** (Data Delete Service): ☒ 自动创建 (Auto Create) ☐ 使用已有服务 (Use Existing Service) [Input] 选择... [Button]

At the bottom right, there are two buttons: "下一步" (Next Step) and "取消" (Cancel).

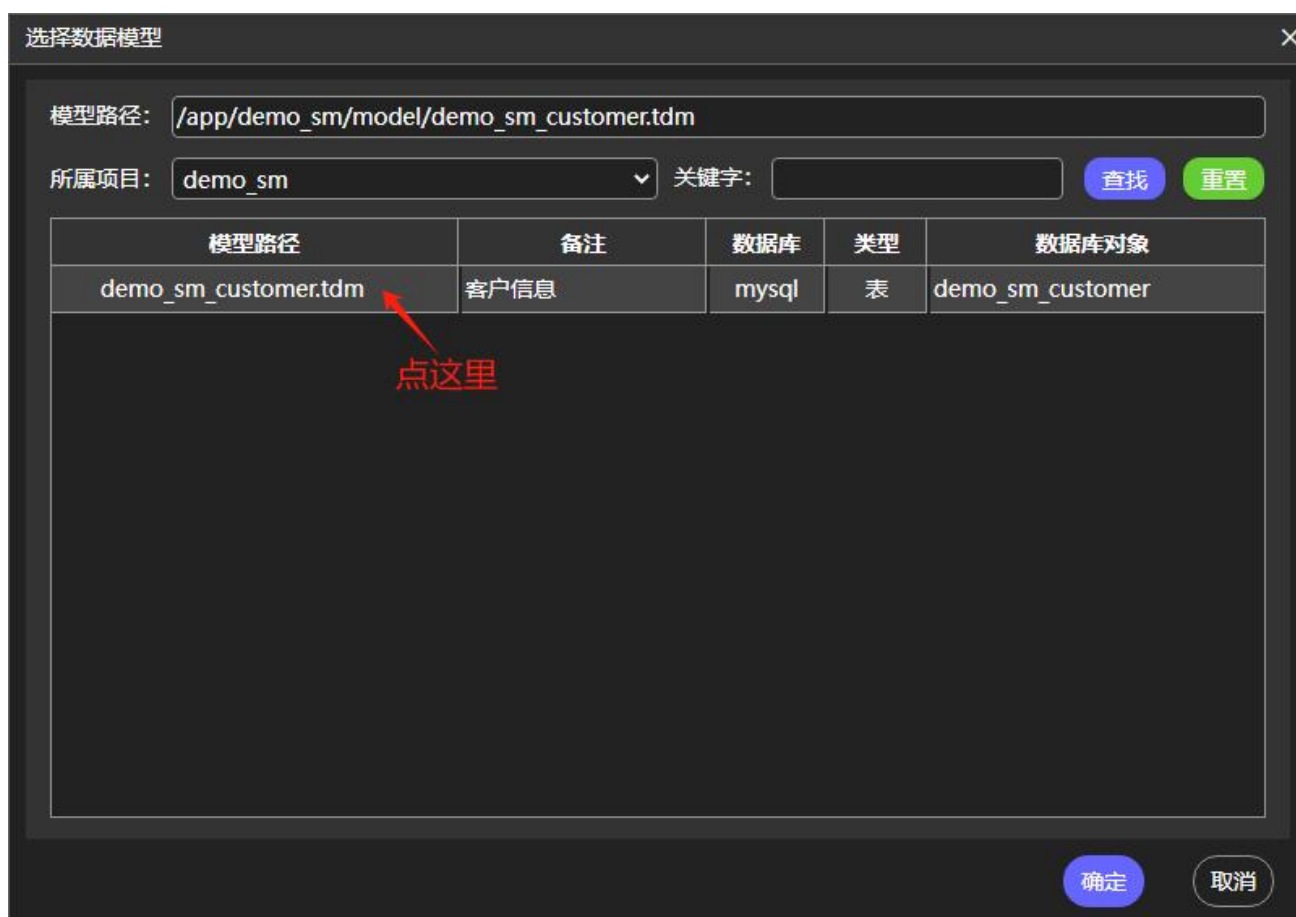
另外，还可以设置以下参数：



- **是否启用翻页：**默认启用，如果数据比较多，建议翻页，否则页面打开会很慢，还会造成服务器卡顿；
- **每页最多显示数据记录条数：**只有启用翻页时，该参数才有效，默认为 20 条；
- **最多查询数据记录条数：**用来设置整个查询最多查多少条记录，默认不限制；
- **是否去掉重复项：**如果选中，则有多条记录所有字段的值都一致时，仅返回一条记录；
- **显示添加按钮：**如果选中，则会在页面顶部工具栏显示添加按钮，点击后可以打开数据添加页面，该选项默认选中。在该选项后面，可以设置添加页面的文件名（默认为 add.tfp）和页面类型（默认为抽屉页）；
- **显示修改按钮：**如果选中，则会在列表内每行数据的最右侧显示修改按钮，点击后可以打开数据修改页面，该选项默认选中。在该选项后面，可以设置修改页面的文件名（默认为 update.tfp）和页面类型（默认为抽屉页）；
- **显示删除按钮：**如果选中，则会在列表内每行数据的最右侧显示删除按钮，点击后可以删除该行数据，该选项默认选中；
- **查询条件显示在页面左侧：**如果选中该选项，则在该向导后续操作中设置的查询条件输入项会显示在页面左侧，如果不选中该选项，则默认会显示在数据列表上部的面板组件里，默认为不选中。如果查询条件比较多，建议选中该选项，否则查询条件所在面板组件可能会出现横向滚动条，影响美观和操作；
- **数据查询服务相关设置：**用来设置使用数据查询后台服务的方式，可以选择是自动创建还是使用已有的服务，如果是自动创建，则可以设置后台服务返回的包含查询结果的响应参数名，默认为 data；如果使用已有服务，则需要点击右侧的“选择...”按钮，选择要使用的后台服务；默认为自动创建；
- **数据删除服务相关设置：**用来设置使用数据删除后台服务的方式，可以选择是自动创建还

是使用已有的服务，如果使用已有服务，则需要点击右侧的“选择...”按钮，选择要使用的后台服务；默认为自动创建；

在上面的“创建数据查询 TFP”对话框中，点击右上角数据模型输入框右侧的“选择...”按钮，会弹出如下图所示的“选择数据模型”对话框：



在该对话框中，默认会列出当前项目的所有数据模型，现在里面就只有一个我们前面创建的客户信息的数据模型（demo\_sm\_customer.tdm），点击所属项目列表可以查看当前引擎服务器上的所有项目，选中某个项目可以在下面的模型列表中显示这个项目中的所有数据模型，如果模型比较多，在关键字输入框中输入搜索关键字，点击“查找”按钮，可以查找符合条件的数据模型。

我们这里不需要切换项目，直接在当前项目的数据模型列表里点击客户信息这个数据模型，然后点“确定”按钮即可。

然后，在“创建数据查询 TFP”对话框中，将页面标题设置为：客户信息管理，选中“查询条件显示在页面左侧”，其他选项都默认即可，不用修改，然后点击“下一步”按钮，可以看到如下图所示的查询字段设置界面：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段 添加关联字段

| ■ | 字段名称        | 字段备注  | 字段格式(双击编辑)  | 关联数据字典(双击选择) | 字段别名 |
|---|-------------|-------|-------------|--------------|------|
| ■ | id          | 编号    | id          |              |      |
| ■ | code        | 编码    | code        |              |      |
| ■ | name        | 名称    | name        |              |      |
| ■ | creator_id  | 创建人编号 | creator_id  |              |      |
| ■ | create_time | 创建时间  | create_time |              |      |
| ■ | status      | 状态    | status      |              |      |
| ■ | salesman_id | 销售员编号 | salesman_id |              |      |
| ■ | contacts    | 联系人   | contacts    |              |      |
| ■ | telephone   | 联系电话  | telephone   |              |      |
| ■ | scale       | 客户规模  | scale       |              |      |
| ■ | grade       | 客户等级  | grade       |              |      |
| ■ | source      | 客户来源  | source      |              |      |

上一步 下一步 取消

### 3.4.3 设置要查询的字段

#### 3.4.3.1 选择要查询的字段

在上图所示的字段列表中选择当前查询页面需要用到的字段，这些字段包括：

- (1) **主键字段**：例如 id，后续在实现修改或删除功能时，都需要传递主键字段的值来判断具体要对哪条记录进行修改或删除操作；
- (2) **要在数据表格里显示的字段**：对于客户信息来说，需要显示的字段包括 name、create\_time、status、contacts、telephone、scale、grade、source、trade；
- (3) **关联字段**：需要从别的表关联查询的信息，包括省份名称、城市名称、销售员姓名等，后续会介绍如何实现关联查询；

对于要查询的字段，可以设置以下信息：

### 3.4.3.2 设置字段格式

默认等于字段名，可以使用 SQL 函数或表达式对字段数据进行特殊处理，例如用 DATE\_FORMAT() 函数将日期时间类型的数据转换为字符串，用 LEFT() 函数截取字符串左边指定长度的子字符串等。

双击字段格式输入框，可以打开如下图所示的 SQL 编辑助手对话框：



在该对话框上部，可以切换数据源，然后，在界面左侧，可以点击系统宏变量、当前关联的数据模型、当前数据源中的表和视图，在中间就会列表显示对应的变量或字段信息，双击这些变量或字段，可以将其插入到底部的 SQL 编辑器中。在该对话框右侧，会显示当前数据库常用的函数，鼠标放在某个函数上时，右侧有显示帮助的小图标，点击可以查看函数的使用说明及示例，双击函数名可以将函数表达式插入到底部的 SQL 编辑器中。

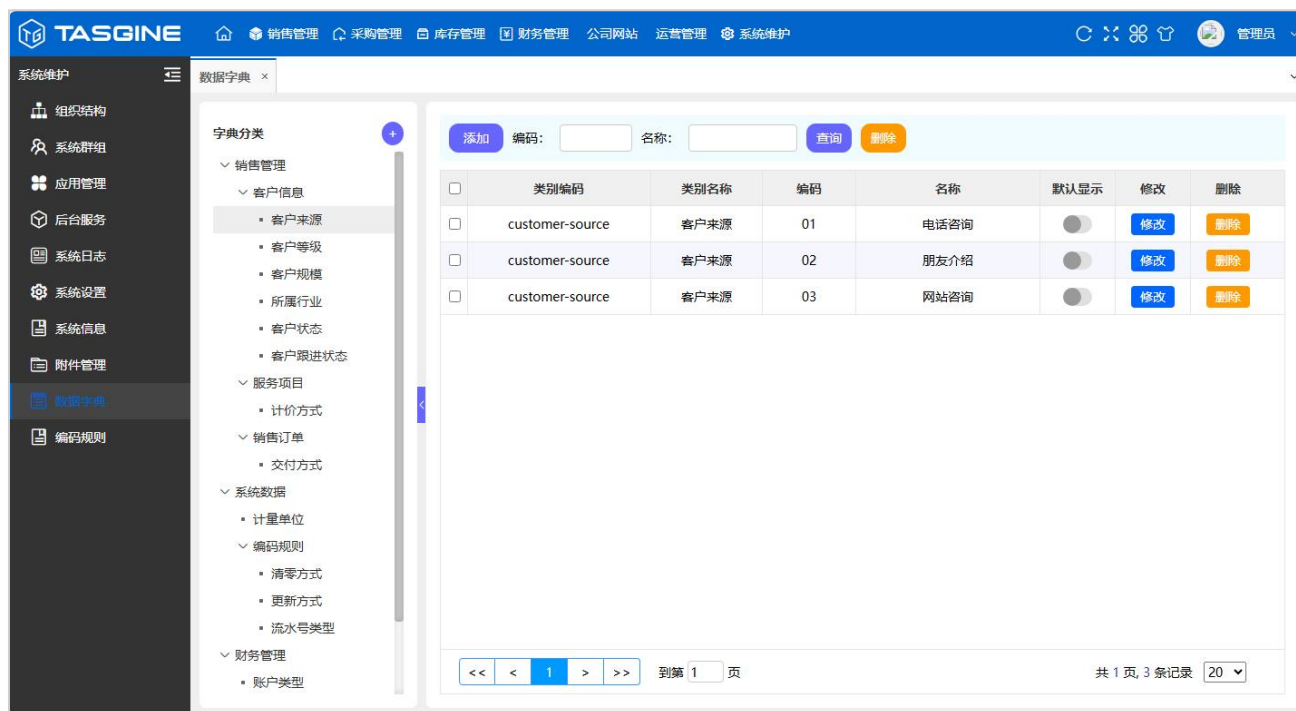
在该对话框底部，可以编写该字段的 SQL 表达式，没有问题后，点“确定”按钮即可。

### 3.4.3.3 设置关联数据字典

如果数据模型某个字段的值有多个固定的可选值，例如客户信息里的客户等级、客户规模、客户来源、所属行业等，建议用任擎后台管理系统中提供的数据字典功能进行统一维护，然后

在添加和修改时，该字段对应的输入项就可以用数据字典组件进行列表显示，供用户选择，否则，如果用下拉列表组件在开发时把选项写死，后续想修改选项就很麻烦。

登录任擎后台管理系统，打开系统维护里的数据字典模块，可以看到如下界面：



在此，可以按照业务类别，在界面左侧的字典分类中，创建数据字典的多级分类，点击某个类别，可以查看里面已经设置好的字典数据，点击上部工具栏面板中的添加按钮，可以打开添加数据字典对话框，如下图所示：

The screenshot shows the '添加数据字典' (Add Data Dictionary) dialog box. It has three input fields: '编码' (Code) with a red asterisk, '名称' (Name) with a red asterisk, and '类别' (Category) with a red asterisk. The '类别' field is a dropdown menu showing '----客户来源'. At the bottom, there are two buttons: '确定' (Confirm) and '取消' (Cancel).

在此添加新的字典数据，从上图可以看到，字典由编码、名称和类别三个字段构成，编码一般用固定位数的数字，例如 01、02、03 等，名称就是具体的选项值，例如客户级别的 A 级、

B 级、C 级等。

点击数据字典列表里各个字典数据右侧的修改和删除按钮,可以对字典数据进行修改和删除操作。

在任擎的安装包里,系统自动创建了一些常用的数据字典,包括销售管理、财务管理和一些系统数据等,我们在开发销售管理这个示例项目时可以直接使用,后续大家在开发自己的各种业务系统时,如果现有的数据字典不够用,可以自己在此创建和维护。

在添加和修改页面里,如果某些字段用的数据字典输入项,则在提交表单后,该字段保存到数据表里的值是字典编码,例如 01、02、03 等,在查询数据时,需要关联数据字典查询对应的选项名称,例如客户信息里的 scale (客户规模) 这个字段,操作方法如下:

在上面的设置查询字段的界面中,找到 scale 这个字段,在它所在行的“关联数据字典”输入框中双击鼠标,可以打开选择数据字典的对话框,如下图所示:



在该对话框中，用鼠标双击销售管理->客户信息中的客户规模，即可完成选择。

对于客户信息来说，其中客户状态、客户规模（scale）、客户等级（grade）、客户来源（source）和所属行业（trade）这几个字段都可以使用数据字典来实现关联查询。

#### 3.4.3.4 设置字段别名

如果为字段设置了特殊的字段格式或关联的数据字典，则可以设置该字段的别名，以便跟原有的字段名称进行区分，字段别名只能是字母、数字或下划线，且必须以字母开头，如果字段设置了别名，则在后续进行数据绑定时，都需要使用别名，而不是原来的字段名。

另外，在查询字段设置界面，还可以进行以下操作：

#### 3.4.3.5 添加自定义字段

如果数据模型中现有的字段无法满足需求，可以点击查询字段设置界面上部的“添加自定义字段”按钮，就会在字段列表底部增加一个空行，在此可以按照需求设置字段名和字段格式等信息。自定义字段一般都是通过设置字段格式里的 SQL 表达式或子查询语句来实现新增字段，例如将两个字段拼接到一起形成一个新的字段，或者用子查询关联查询其他表里的某个字段等。

#### 3.3.3.6 添加关联字段

按照数据库设计范式的第三范式要求，一个数据库表中不能包含已在其它表中存在的非主关键字信息，例如客户信息里已经包含客户名称、联系人、联系电话等客户信息，则销售订单表里就不需要再定义字段记录这些信息，只需要定义一个客户编号字段，然后用这个字段的值去客户信息表里关联查询客户有关信息即可，在这种情况下，就需要用到关联字段的功能。

当然，这个设计范式不是强制性的，如果某个表里的数据量特别大，而且用户访问特别频繁，也可以适当保留一些冗余字段，尽量减少关联查询，以便提高查询性能。

对于客户信息这个表来说，有以下信息需要关联其他表查询：

| 关联字段别名 | 关联字段备注 | 关联表 | 关联表字段 | 关联条件 |
|--------|--------|-----|-------|------|
|--------|--------|-----|-------|------|



|               |       |                            |      |                                          |
|---------------|-------|----------------------------|------|------------------------------------------|
| salesman_name | 销售员姓名 | sys_user<br>(系统用户表)        | name | id =<br>demo_sm_customer.salesman<br>_id |
| province_name | 省份名称  | sys_data_region<br>(地区编码表) | name | code =<br>demo_sm_customer.province      |
| city_name     | 城市名称  |                            | name | code =<br>demo_sm_customer.city          |

下面以添加客户信息的省份名称字段为例，给大家介绍如何使用关联字段的功能。

在查询字段设置界面上部，点击“添加关联字段”按钮，会弹出如下所示对话框：

在该对话框中，可以设置以下信息：

- **关联对象类型：**包括表和视图，默认为表；
- **关联数据对象：**就是具体的表或视图，这里我们选择系统地区编码表 sys\_data\_region，选完数据对象后，会在中间的关联表字段列表里显示这个表里的所有字段，双击字段名，可以插入到底部的关联条件编辑框中；

- **关联到主表的字段：**用来设置要把字段值关联到主表里作为查询结果返回的字段，这里选择 name 字段，表示要在查询结果里返回系统地区编码表里的地区名称数据；
- **关联字段别名：**用来设置关联数据在最终的查询结果里体现的字段名称，不能跟主表里的字段名重名，因为客户信息表里已经有 name 字段了，所以这里要设置为 province\_name，以便进行区分；
- **关联字段备注：**用来设置关联字段的备注信息，非必填，但建议设置，因为后续设置数据绑定时，可以显示该备注作为参考，这里设置为：省份名称；
- **关联条件：**用来设置主表和关联表之间的关联关系，点击上面关联表字段列表或主表字段列表里的具体字段，可以将字段名插入到关联条件编辑框中，另外，在编辑框上部还提供了常用的运算符，点击即可插入。这里我们设置为：`code=demo_sm_customer.province`，表示这两个表的关联关系为系统地区编码表里的编码字段（code）的值等于客户信息表里的省份信息（province）字段的值。

上述内容都设置好之后，点“确定”按钮，即可将该关联字段的信息添加到主表的查询字段列表中。

城市名称（city\_name）和销售员姓名（salesman\_name）这两个关联字段的添加就不再详细讲解，可以参考上面的操作。

客户信息查询字段设置的最终结果如下图所示：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段

添加关联字段

|                                     | 字段名称        | 字段备注  | 字段格式(双击编辑)  | 关联数据字典(双击选择)    | 字段别名 |
|-------------------------------------|-------------|-------|-------------|-----------------|------|
| <input checked="" type="checkbox"/> | id          | 编号    | id          |                 |      |
| <input checked="" type="checkbox"/> | code        | 编码    | code        |                 |      |
| <input checked="" type="checkbox"/> | name        | 名称    | name        |                 |      |
| <input type="checkbox"/>            | creator_id  | 创建人编号 | creator_id  |                 |      |
| <input checked="" type="checkbox"/> | create_time | 创建时间  | create_time |                 |      |
| <input checked="" type="checkbox"/> | status      | 状态    | status      | customer-status |      |
| <input type="checkbox"/>            | salesman_id | 销售员编号 | salesman_id |                 |      |
| <input checked="" type="checkbox"/> | contacts    | 联系人   | contacts    |                 |      |
| <input checked="" type="checkbox"/> | telephone   | 联系电话  | telephone   |                 |      |
| <input checked="" type="checkbox"/> | scale       | 客户规模  | scale       | customer-scale  |      |
| <input checked="" type="checkbox"/> | grade       | 客户等级  | grade       | customer-grade  |      |
| <input checked="" type="checkbox"/> | source      | 客户来源  | source      | customer-source |      |

上一步

下一步

取消

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段

添加关联字段

|                                     | 字段名称          | 字段备注  | 字段格式(双击编辑)                                    | 关联数据字典(双击选择)    | 字段别名          |
|-------------------------------------|---------------|-------|-----------------------------------------------|-----------------|---------------|
| <input checked="" type="checkbox"/> | source        | 客户来源  | source                                        | customer-source |               |
| <input checked="" type="checkbox"/> | trade         | 所属行业  | trade                                         |                 |               |
| <input type="checkbox"/>            | site          | 客户网址  | site                                          |                 |               |
| <input type="checkbox"/>            | province      | 所在省份  | province                                      |                 |               |
| <input type="checkbox"/>            | city          | 所在城市  | city                                          |                 |               |
| <input type="checkbox"/>            | address       | 详细地址  | address                                       |                 |               |
| <input type="checkbox"/>            | comment       | 备注    | comment                                       |                 |               |
| <input type="checkbox"/>            | deletor_id    | 删除人编号 | deletor_id                                    |                 |               |
| <input type="checkbox"/>            | delete_time   | 删除时间  | delete_time                                   |                 |               |
| <input checked="" type="checkbox"/> | salesman_name | 销售员姓名 | (select name from sys_user where id = demo_si |                 | salesman_name |
| <input checked="" type="checkbox"/> | province_name | 省份名称  | (select name from sys_data_region where code  |                 | province_name |
| <input checked="" type="checkbox"/> | city_name     | 城市名称  | (select name from sys_data_region where code  |                 | city_name     |

上一步

下一步

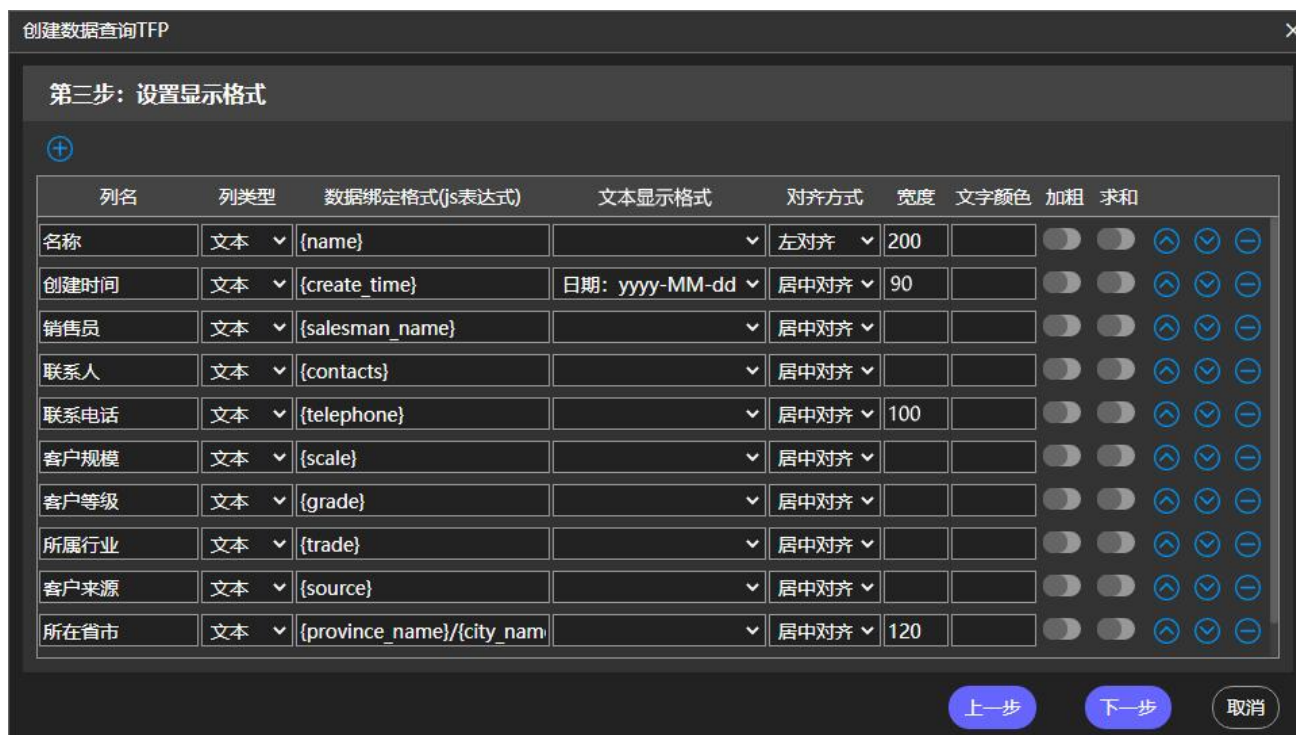
取消

设置好查询字段后，点下一步按钮，打开“设置显示格式”界面。

3.4.4 设置数据列表显示格式

数据列表里显示的内容不一定完全是按照查询设置里的字段里显示，例如主键 id 字段就不需要显示，数据列表里的有些列可能是由多个字段拼接接到，例如所属省市这一列的数据就是

由省份名称和城市名称拼接到一起的，所以，在创建数据查询列表页面时，除了前面设置的查询字段，还要设置数据表格各个列的显示格式，设置界面如下图所示：



该列表里默认会显示第二步选择的所有查询字段，不需要显示的可以删除，需要合并可以合并，而且，可以点击右侧的上移和下移按钮修改字段的显示顺序。对于本示例来说，客户信息最终设置的显示格式如上图所示，另外，还有一个名为“客户状态”的列没显示出来，其数据绑定格式为：{status}，其他选项都是默认。

在上图的界面里，可以为数据表格各个列设置以下信息：

- **列名**：用来设置列的标题；
- **列类型**：用来设置列的内容类型，可以选择文本、按钮、链接、图片、开关、进度条和自定义，默认为文本；
- **数据绑定格式**：用来设置这一列的数据绑定格式，默认为：**{字段名}**，表示这一列的内容为上一步里设置的指定查询字段的值，多个字段值可以拼接，也可以加入静态字符，例如：

**{province\_name}/{city\_name}**；

- **文本显示格式**：用来设置文本类型的字段常用的一些显示格式，包括金额、文件大小、年月：yyyy-MM、年月：yyyy/MM、日期：yyyy-MM-dd、日期：yyyy/MM/dd、日期：yyyy 年 MM 月 dd 日这几种可以选择；
- **对齐方式**：用来设置该列数据的水平对齐方式，默认居中对齐，如果该列内容的长度不固定，有的比较长，有的比较短，这建议设置为左对齐，如果是金额建议设置为右对齐；
- **宽度**：用来设置该列的宽度，默认为 80 像素，可以根据需要设置；
- **文字颜色**：用来设置该列文字的颜色；
- **加粗**：用来设置该列文字是否需要加粗，默认不需要；
- **求和**：用来设置该列是否需要计算合计，默认不需要；

都设置好之后，点击“下一步”按钮，可以进入第四步的设置。

### 3.4.5 设置动态查询参数

一个功能完备的数据列表页面应该给用户提供丰富的查询条件，以便用户可以根据各种条件查询自己想要的数据，为此，TaskBuilder 的创建数据查询页面的向导提供了专门的设置动态查询参数的界面，所谓“动态查询参数”是指在进行数据查询时，如果前端页面给后台服务传了该参数，就使用该参数对应的查询表达式作为查询条件进行查询，如果该参数没有传值，则不使用该参数对应的查询表达式作为查询条件进行查询。多个参数都有值时，它们对应的多个查询条件是并且（and）的关系，即只查询这些条件都满足的数据记录。

在上图所示界面中，先在上部的查询字段列表里选择要作为查询条件的字段，然后在后面的查询条件列表里选择查询条件，然后点最后面的加号图标，即可添加一个动态查询参数。

可选的查询条件包括以下几种：

- **模糊查找**：表示查找指定字段的值包含指定查询参数的数据记录，该查询条件默认的查询表达式是：{%req.查询参数名%}，表示只要该条件对应的字段值包含指定查询参数的值，

就算是符合条件；如果只想查询前缀为指定查询参数值的记录，则可以将查询表达式设置为：`{req.查询参数名%}`；反之，如果只想查询后缀为指定查询参数值的记录，则可以将查询表达式设置为：`{%req.查询参数名}`；这里的%是 SQL 中的 **通配符**，表示匹配 0 个或多个字符。模糊查询只支持字符串类型的字段，数值、日期等类型不支持使用模糊查询。

- **等于**：表示查找指定字段的值与指定查询参数相等的数据记录，例如：`id={req.id}`，表示要查询 id 字段的值等于名为 id 的查询参数的值。等于判断可以用于数值、日期时间、字符、短字符串等类型的字段，字符串类型的字段如果有比较长的字符串值，不建议进行等于判断；文本、二级制类型的字段不能用等于判断。
- **大于**：表示查找指定字段的值大于指定查询参数的数据记录，例如：`id>{req.id}`，表示要查询 id 字段的值大于名为 id 的查询参数的数据记录。大于判断可用于数值和日期时间类型的字段，不能用于字符、字符串、二进制类型的字段。
- **小于**：表示查找指定字段的值小于指定查询参数的数据记录，例如：`id<{req.id}`，表示要查询 id 字段的值小于名为 id 的查询参数的数据记录。小于判断可用于数值和日期时间类型的字段，不能用于字符、字符串、二进制类型的字段。
- **大于或等于**：表示查找指定字段的值大于或等于指定查询参数的数据记录，例如：`id>={req.id}`，表示要查询 id 字段的值大于或等于名为 id 的查询参数的数据记录。大于或等于判断可用于数值和日期时间类型的字段，不能用于字符、字符串、二进制类型。
- **小于或等于**：表示查找指定字段的值小于或等于指定查询参数的数据记录，例如：`id<={req.id}`，表示要查询 id 字段的值小于或等于名为 id 的查询参数的数据记录。小于或等于判断可用于数值和日期时间类型的字段，不能用于字符、字符串、二进制类型。
- **不等于**：表示查找指定字段的值不等于指定查询参数的数据记录，例如：`id<>{req.id}`，表示要查询 id 字段的值不等于名为 id 的查询参数的数据记录。不等于判断可用于数值、日期时间、字符、短字符串等类型的字段，字符串类型的字段如果有比较长的字符串值，不建议进行不等于判断；文本、二级制类型的字段不能用不等于判断。
- **起止日期**：用来同时添加开始日期和结束日期这两个查询参数，仅能用于日期时间或日期类型的字段，如果查询字段的类型是日期时间格式 (yyyy-MM-dd HH:mm:ss)，则在开始日期的查询表达式里，需要加上 0:00:00，结束日期的查询表达式里，需要加上 23:59:59，否则查询结果会不准确，因为前端页面传给后台服务的起止日期只有日期值，不包含时间。



添加后的动态查询参数可以设置以下几项内容：

- **参数名称：**用来设置从列表页面传递给后台服务的查询参数的名称，不一定非得跟数据模型中的字段名一致，对应的是列表页面里查询条件输入项组件的编号。如果设置了多个查询参数，则这些查询参数的名称不能重复；
- **参数说明：**用来设置列表页面里，各个查询条件输入项的文本标签的内容；
- **输入项类型：**用来设置列表页面里，各个查询条件输入项的类型，默认为单行文本输入框；
- **查询表达式：**用来设置在数据库中执行实际查询时，各个查询参数对应的查询条件表达式，在添加查询参数时会根据选择的查询条件自动生成，也可以手动输入或修改，双击查询条件表达式输入框，可以打开 SQL 编辑助手编写复杂的查询条件表达式；
- **默认值：**用来设置在列表页面里，该查询条件输入项的默认值，一般只有单选框、多选框和下拉列表这几种类型的输入项需要设置默认值；
- **数据字典：**用来设置在列表页面里，该查询条件输入项是否关联指定的数据字典，显示其选项列表，如果设置了数据字典，则输入项类型会自动使用下来列表形式；
- **可选值：**用来设置单选框、多选框和下拉列表这几种输入项的可选值；

创建数据查询TFP

第四步：设置动态查询参数

查询字段：  查询条件：  (+)

| 参数名称        | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)                    | 默认值 | 数据字典(双击选择)      | 可选值 |
|-------------|------|-------|----|--------------------------------|-----|-----------------|-----|
| name        | 客户名称 | 单行文本  |    | name like {%req.name%}         |     |                 |     |
| contacts    | 联系人  | 单行文本  |    | contacts like {%req.contacts%} |     |                 |     |
| scale       | 客户规模 | 下拉列表  |    | scale={req.scale}              |     | customer-scale  |     |
| trade       | 所属行业 | 下拉列表  |    | trade={req.trade}              |     | customer-trade  |     |
| grade       | 客户等级 | 下拉列表  |    | grade={req.grade}              |     | customer-grade  |     |
| source      | 客户来源 | 下拉列表  |    | source={req.source}            |     | customer-source |     |
| salesman_id | 所属销售 | 用户    |    | salesman_id={req.salesman_id}  |     |                 |     |
| status      | 状态   | 下拉列表  |    | status={req.status}            |     | customer-status |     |

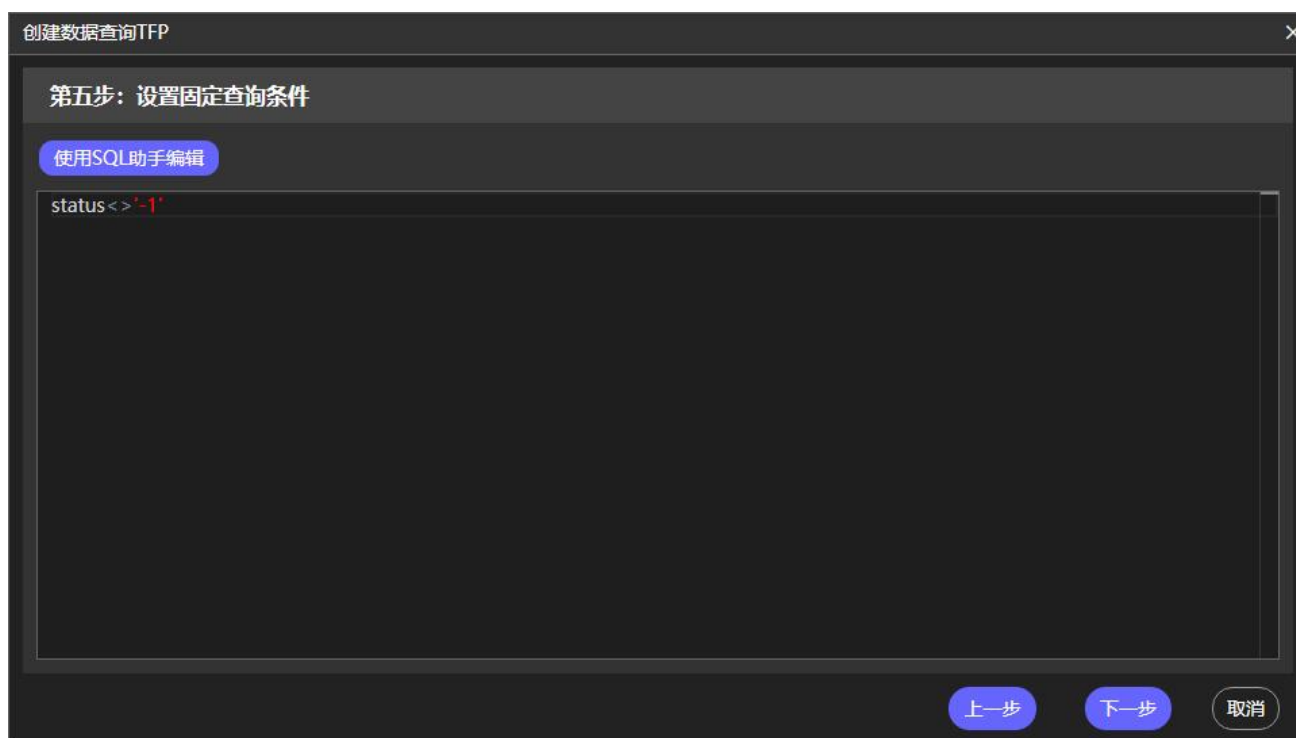
上一步 下一步 取消



对于我们要创建的客户信息列表页面来说,可以参考上图设置相应的动态查询参数,设置好后,点击下一步按钮,即可进行第五步的操作。

### 3.4.6 设置固定查询参数

在“创建数据查询TFP”向导的第五步,可以设置固定查询参数,“固定”是指不管用户怎么查,这些条件都需要满足,对于某些列表页面,如果需要设置这种固定的查询条件,则可以在此进行设置,例如,我们要创建的客户信息列表页面,不需要显示状态为已删除(status='1')的客户信息,则可以在此将固定查询条件设置为: status<>'1' , 设置界面如下图所示:



然后继续点击下一步, 进入第六步的设置。

### 3.4.7 设置排序条件

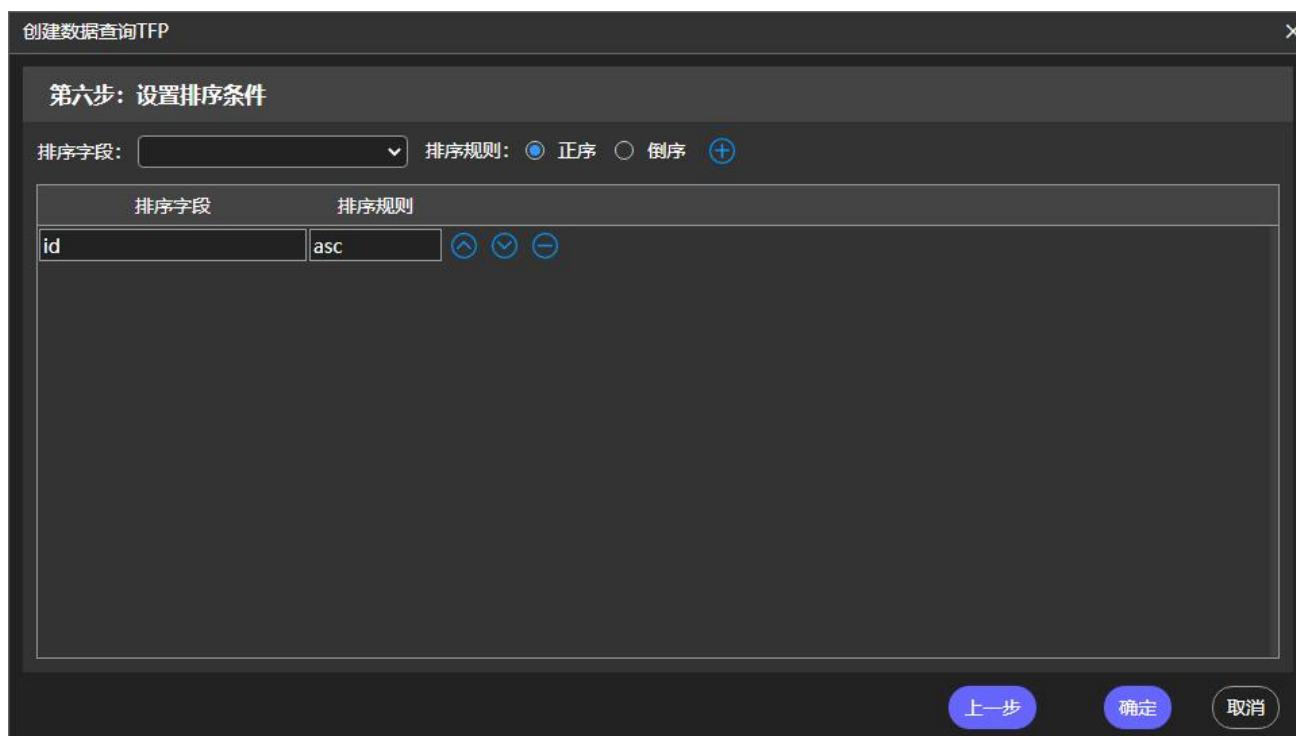
在整个向导的最后一步,可以设置是否按指定字段的值对查询结果进行排序,支持正序和倒序两种排序方式。如果没有设置任何排序字段,则默认按数据库里现有数据记录的实际存储的先后顺序排序。如果设置了多个排序条件,则按这些条件从上到下的顺序按照先后进行排序。

不同类型的数据，正序和倒序的排序规则见下表：

| 字段数据类型        | 正序          | 倒序          |
|---------------|-------------|-------------|
| 数值，包括整数、小数、金额 | 按数值从小到大排序   | 按数值从大到小排序   |
| 日期时间          | 按日期时间从前往后排序 | 按日期时间从后往前排序 |
| 英文字符串         | 按字母表从前往后排序  | 按字母表从后往前排序  |
| 汉字字符串         | 按拼音从前往后排序   | 按拼音从后往前排序   |

一般来说，像客户信息这种基础资料，按默认的增加顺序排序即可，而对于销售订单、销售收款记录等这种单据类的列表页面，如果是列表显示所有客户的单据，那么建议按下单时间和单价编号倒序排列，最后下单的单据显示在最前面，这样单据多了后，可以优先看到最新的。

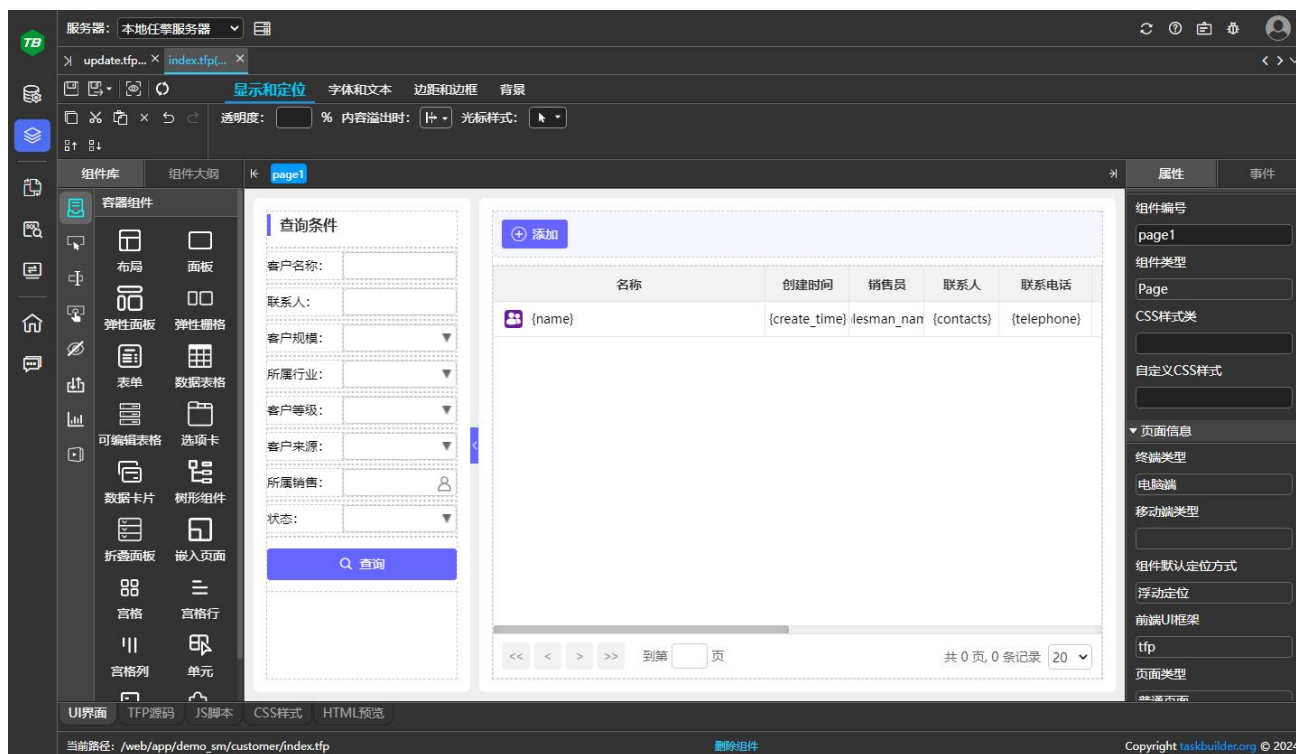
排序条件设置界面如下图，在界面上部的排序字段里选择需要排序的字段，然后在右侧选择排序规则，然后点加号，即可添加对应的排序条件，这里为了演示，添加了一个按客户编号(id) 字段正序排序的规则，其实，默认就是按该规则排序，不需要特地设置。



最后，点击确定按钮，即可完成客户信息列表页面的创建。

### 3.4.8 查看和预览客户信息列表页面

用向导创建完页面后，TaskBuilder 会自动用前端页面设计器打开，界面如下图所示：



在此可以继续修改完善该页面，选择各个组件，可以设置其样式、属性和事件，可以删除不要的组件，可以从左侧组件库中拖拽新的组件。目前这个客户列表页面暂时不需要改动。

点击前端页面设计器工具栏上的预览按钮会打开前端页面预览工具：



这时，预览工具里可能会出现一片空白，不要着急，请稍等片刻，这是因为任擎正在编译该 tfp 页面，我们在任擎服务管理器里可以看到如下图所示的日志：

```
16:19:17.229 rid:2407131619170012 请求路径: /app/demo_sm/customer/index.tfp?_isDebug=true
16:19:17.266 正在生成/app/demo_sm/customer/index.tfp的js代码...
16:19:17.266 正在将/app/demo_sm/customer/index.tfp的js代码转换为ES5格式...
Browserslist: caniuse-lite is outdated. Please run:
  npx update-browserslist-db@latest
  Why you should do it regularly: https://github.com/browserslist/update-db#readme
16:19:24.104 正在打包/app/demo_sm/customer/index.tfp的js代码...
16:19:27.137 正在生成/app/demo_sm/customer/index.tfp的HTML代码...
```

预览 tfp 页面时，为了保证看到的是最新的内容，每次都会进行编译，在正式环境中，tfp 页面只需要编译一次就会自动生成其对应的 HTML 文件，后续再有用户访问，会自动加载该 HTML 文件的内容，除非该 tfp 页面又进行了修改，否则不会每次有人访问都编译。

编译完毕后，就会在预览工具中显示该页面实际运行后的界面效果，如下图所示：



由于我们还没有添加客户信息，所以预览时，界面里的客户信息列表是空白的，下面将带领大家实现添加客户信息的功能。

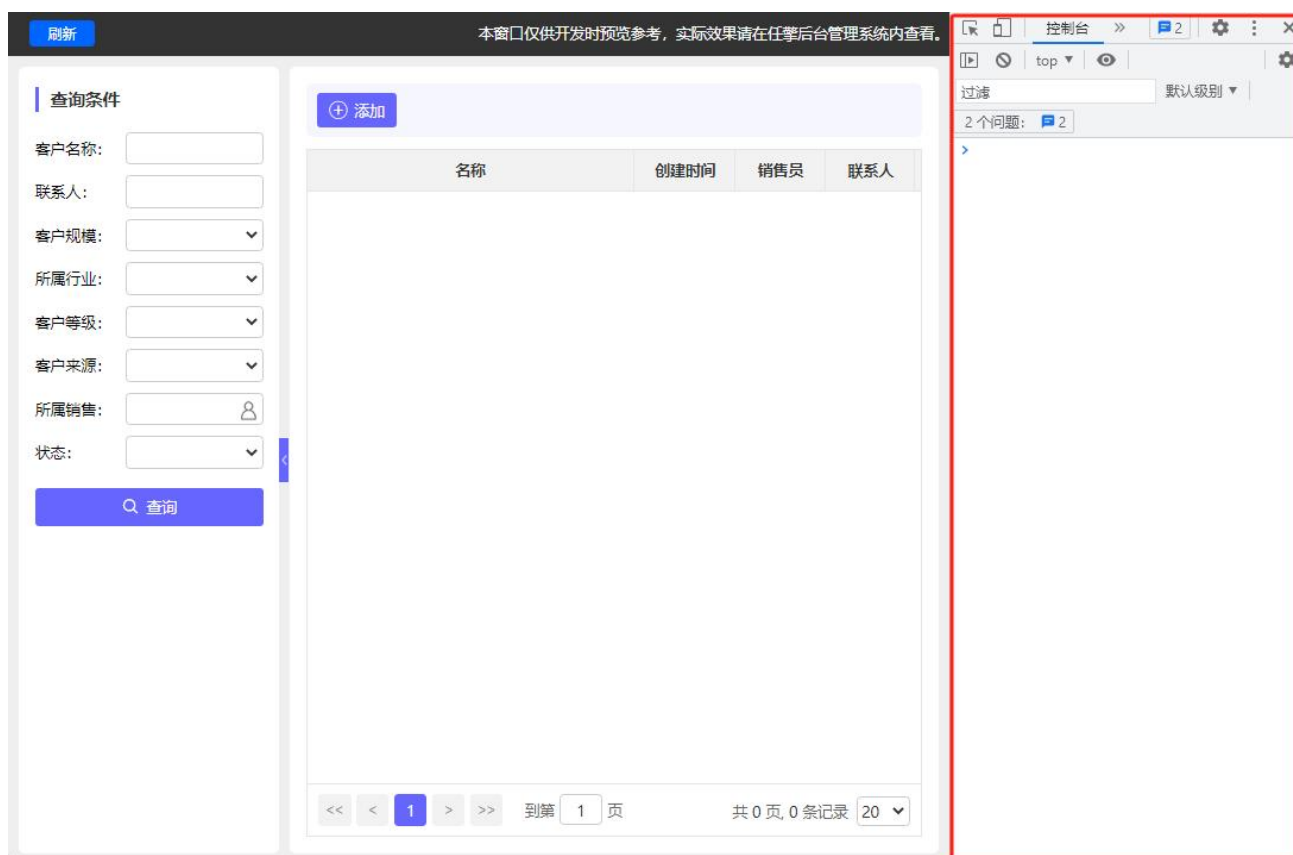
### 3.5 如何查看前后端错误和调试信息

在继续开发后续功能之前，给大家介绍一下如何查看前后端错误和调试信息，因为我们在实际的开发过程中，不会总是一帆风顺，可能会经常出现各种问题，这时候如果要判断具体是什么原因造成的，则前后端的调试信息就非常重要。

## 3.5.1 查看前端控制台信息

### 3.5.1.1 在预览窗口中查看控制台信息

在 TaskBuilder 的前端页面设计器里，点击工具栏上的“预览”按钮，会打开预览窗口，在该窗口右侧默认会显示一个控制台界面，如果当前预览的页面出现错误，在此会显示具体的错误信息，另外，我们也可以手动输出一些调试信息，以便检查页面里的脚本是否正确。

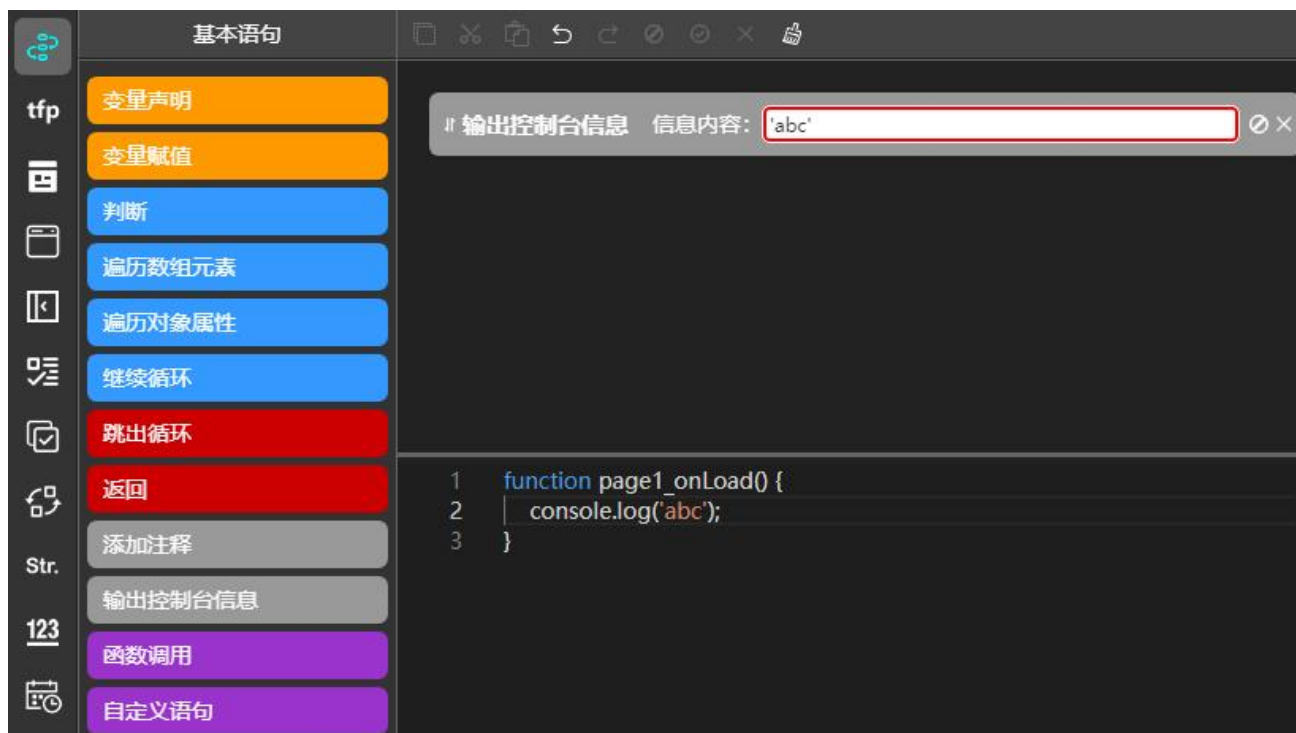


下面，我们人为制造一个错误，来看一下效果：

在 TaskBuilder 里打开前面创建的客户信息列表页面，选中页面组件（page1），然后在右侧的事件设置面板里，双击“页面加载后”事件，打开事件函数设置界面，从基本语句里拖拽一条“输出控制台信息”语句到右侧编辑区，然后输入信息内容：abc，然后保存页面，打开预览工具（如果已经打开，点刷新即可）预览该页面，就可以在右侧控制台看到如下错误信息：



该错误信息的第一句说明了错误的具体原因：page1\_onLoad 函数里的 abc 这个变量没有定义（如果英文不太好，可以将错误提示复制到翻译软件内翻译），因为“输出控制台信息”语句的内容必须是有效的 JavaScript 值或表达式，而 abc 这个字符既不是变量名，也不是有效的字符串值，如果要想将 abc 作为字符串输出，需要在两边加上单引号，如下图所示：

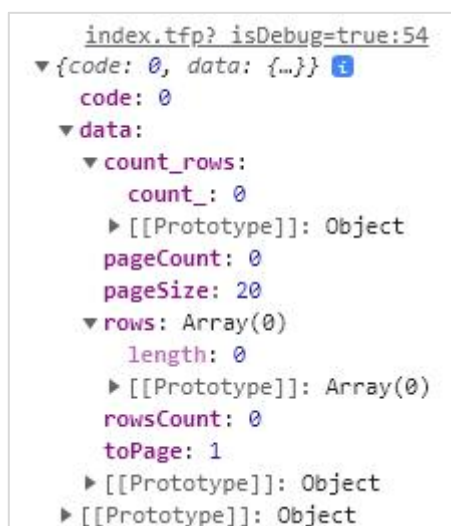


这时，我们再刷新预览窗口，就能看到正确的输出，如下图：





我们再做个例子，在前端页面设计器中选中客户信息列表页面里的数据表格组件 grid1，然后双击右边事件设置面板里的“加载数据后”事件输入框，进入该事件的编辑界面，从基本语句里拖拽一条“输出控制台信息”语句到右边编辑区，然后输入内容：res，然后保存页面，并刷新预览窗口，就可以看到控制台的输出内容如下图所示：



上面输出的这个 res 变量是数据表格组件的“加载数据后”事件函数传入的一个参数，里面包含了后台服务返回给前端页面的所有数据，我们在开发时，如果不确定后台服务是否返回了正确的数据，就可以在控制台中输出 res 的内容，进行检查和判断。

### 3.5.1.2 禁用预览窗口的缓存功能

在开发时，我们有时会发现在前端页面设计器里修改了内容，但在预览时这些内容并没有发生变化，这可能是因为预览窗口的缓存机制造成的，您看到的内容有可能是上一次打开页面时缓存的内容，为了避免出现这种情况，建议将预览窗口的缓存功能禁用，操作方法如下：

点击预览窗口右侧调试栏顶部的 ，在弹出的菜单里选择“网络（Network）”，然



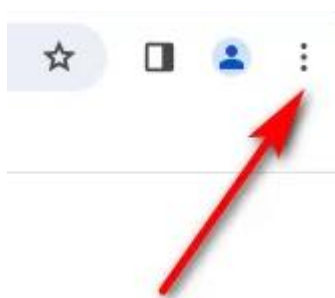
后将下面的“停用缓存（Disable cache）”选项打上勾，这样再打开预览窗口查看页面时，每次都会看到最新的内容。



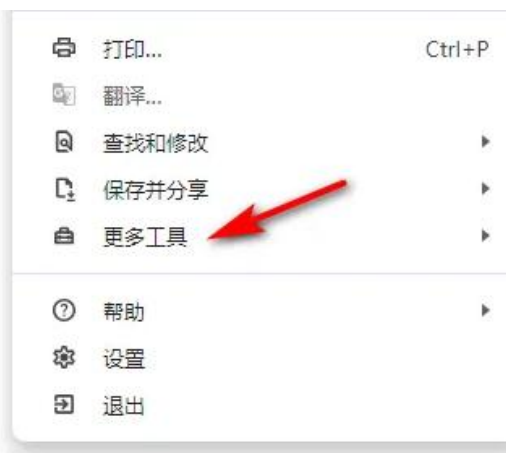
### 3.5.1.3 在浏览器里查看控制台信息

如果我们开发的页面已经发布到后台管理系统里，能够用浏览器查看，这时如果页面出现了问题，需要查看错误或调试信息，可以按 F12 快捷键，打开浏览器的调试窗口，或者在浏览器的设置菜单里，打开开发者工具，下面分别以 Chrome 浏览器为例，给大家介绍如何打开。

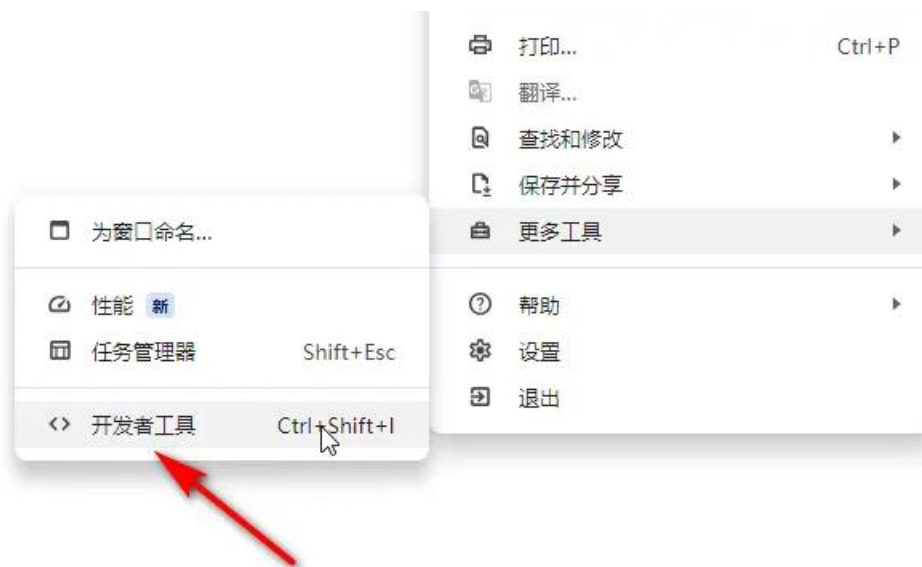
(1) 首先点击浏览器右上角的更多按钮，如下图所示：



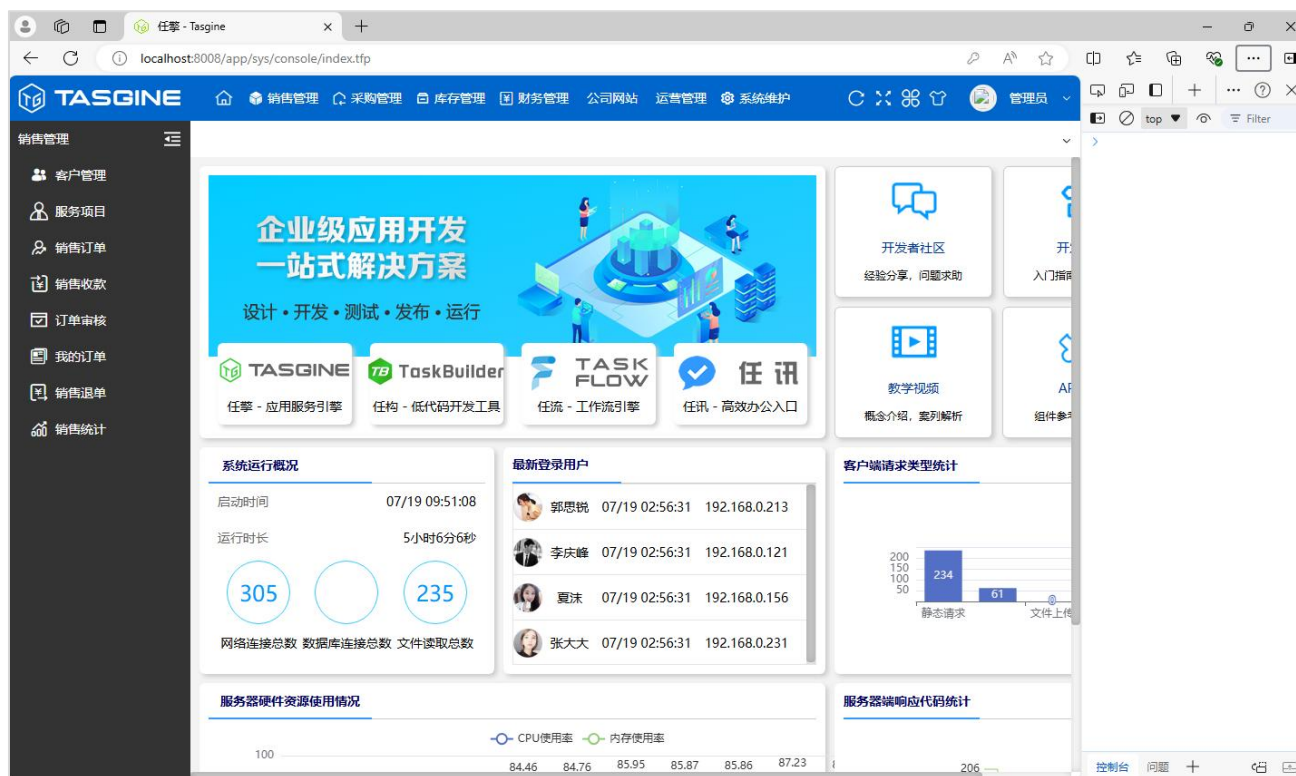
(2) 在弹出的菜单中选择“更多工具”，如下图所示：



(3) 然后再点击弹出的子菜单中的“开发者工具”，如下图所示：



然后就可以在浏览器右侧看到控制台窗口了，如下图所示：



在浏览器的控制台中输出信息的方法和上面的预览窗口是一样的，这里就不再举例说明。

### 3.5.2 查看后台服务的输出信息

在客户端请求后台服务时，会输出很多日志信息，这些信息对于开发调试来说非常有用，包括客户端发给服务器的详细请求数据、数据库操作最终执行的 SQL 语句及参数、后台服务返

回给前端的详细数据、后台服务执行过程中出现的错误信息等等，当我们开发的应用出现问题时，一般通过查看日志就能大致判断出来问题出现在哪个环节，是客户端没有传正确的参数，还是服务器端执行出错，或者返回的数据不对等等，学会查看日志是每个开发者必须具备的基本技能，可以通过以下几种方式查看后台服务输出的信息，下面分别介绍一下：

### 3.5.2.1 通过任擎服务管理器查看

为了方便大家在 Windows 系统内管理任擎服务器，以及查看服务器端的日志信息，我们提供了一个“任擎服务管理器”程序，文件名为“ServerManager.exe”，双击该文件即可打开，界面如下所示：



从上图可以看到，整个界面分为以下几个区域：

- (1) **工具栏**：在此提供了启动或停止选中服务器的按钮，清空日志的按钮和退出按钮；
- (2) **服务器列表**：在此列出了该软件管理的服务器程序，默认包括一个绿色版的 MySQL 数据库软件、任擎主服务器和任擎文件服务器。其中，任擎服务器是必须要启动的，而 MySQL 数

数据库和文件服务器不是必须要启动的。如果数据库要连自己安装的数据库，则这里的 MySQL 数据库就无需启动，文件服务器是给任讯客户端用来发送和接收文件的，如果不用任讯，则该文件服务器也无需启动。选中某个服务器后，如果该服务器没启动，则工具栏上的第一个按钮会显示为“启动”，如果服务器已启动，这个按钮就会显示为“停止”，而且服务器名称后面会显示这个服务器对应的系统进程编号；

(3) **日志输出区**：在此可以查看指定服务器的日志输出，点击上部的选项卡，可以切换具体要看哪个服务器的；对于我们日常开发来说，经常要看的就是任擎服务器的日志，日志内容支持复制；

(4) **状态栏**：这里显示了该服务管理器所在的文件夹路径，如果启动了多个服务管理器，可以通过查看这里的路径进行区分。

### 3.5.2.2 通过服务器控制台查看

在 Linux 和 MacOS 系统内，或者 Docker 环境，无法运行上面介绍的图形化的任擎服务管理器软件，只能在终端控制台里查看服务器日志，界面如下所示：

```
17:04:50.761 [m] [pid:1896]执行语句已成功!
17:04:50.761 [m] [pid:1896]正在执行语句...
17:04:50.761 [m] [pid:1896]语句: select org_id,user_id,dep_id from sys_user_subdep where dep_id in (select id from sys_dep where status=0) and user_id in (select id from sys_user where status=0) order by user_id asc,(select code from sys_dep where id=sys_user_subdep.dep_id) asc
17:04:50.762 [m] [pid:1896]执行语句已成功!
17:04:50.762 [m] [pid:1896]正在执行语句...
17:04:50.762 [m] [pid:1896]语句: select org_id,role_id,user_id from sys_role_user where role_id in (select id from sys_role where status=0) and user_id in (select id from sys_user where status=0) order by role_id asc,(select code from sys_user where id=sys_role_user.user_id) asc
17:04:50.762 [m] [pid:1896]执行语句已成功!
17:04:50.762 [m] [pid:1896]正在关闭连接...
17:04:50.763 Cache已加载!
[任擎:16206]正在启动会话管理器...
[任擎:16206]正在启动计划任务管理器...
[任擎:16206]正在启动消息管理器...
[任擎:16206]正在启动消息客户端...
17:04:50.765 正在连接消息推送服务器:[127.0.0.1:8102]...
[任擎:16206]正在启动后台驻留服务...
[任擎:16206]正在加载tfp组件库文件...
17:04:50.768 [m] [pid:1896]关闭连接已成功!
17:04:50.769 连接消息推送服务器失败: connect ECONNREFUSED 127.0.0.1:8102
17:04:50.770 消息推送网络连接已关闭!
17:04:51.39 正在加载tfp组件默认样式...
[任擎:16206]正在启动HTTP Server...
[任擎:16206]http启动网络端口: 8001
17:04:51.44 任擎httpServer已启动!
```



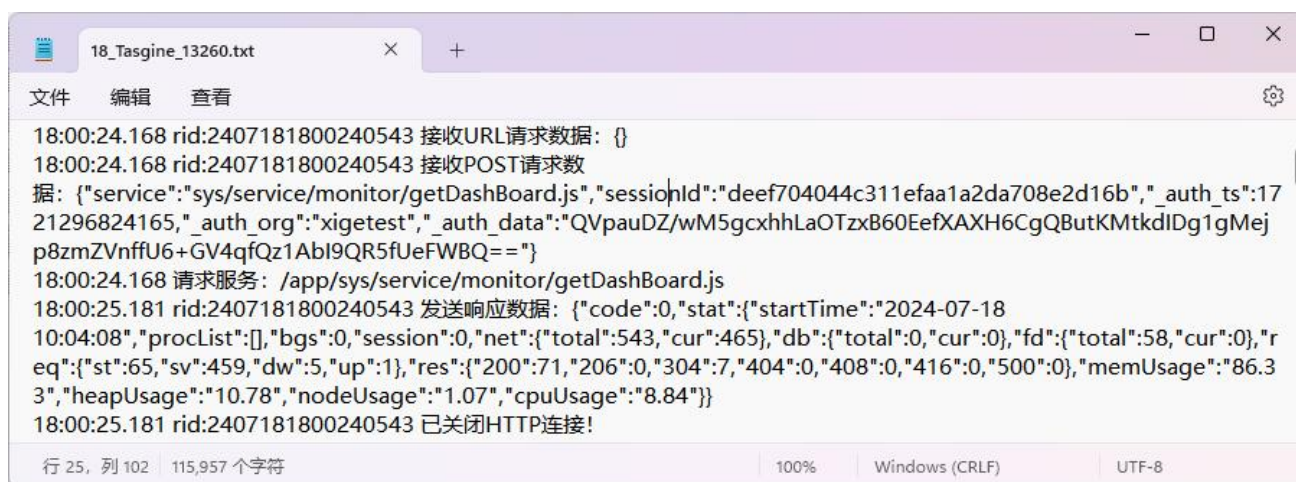
### 3.5.2.3 通过日志文件查看

任擎服务器的日志信息除了输出到控制台，还会记录到日志文件中，保存路径为：

`tasgine\logs\年月日\小时_服务器名称_服务器进程编号.txt`

例如：`tasgine\logs\20240718\10_Tasgine_13280.txt`

打开日志文件，即可查看里面详细的内容，如下图所示：



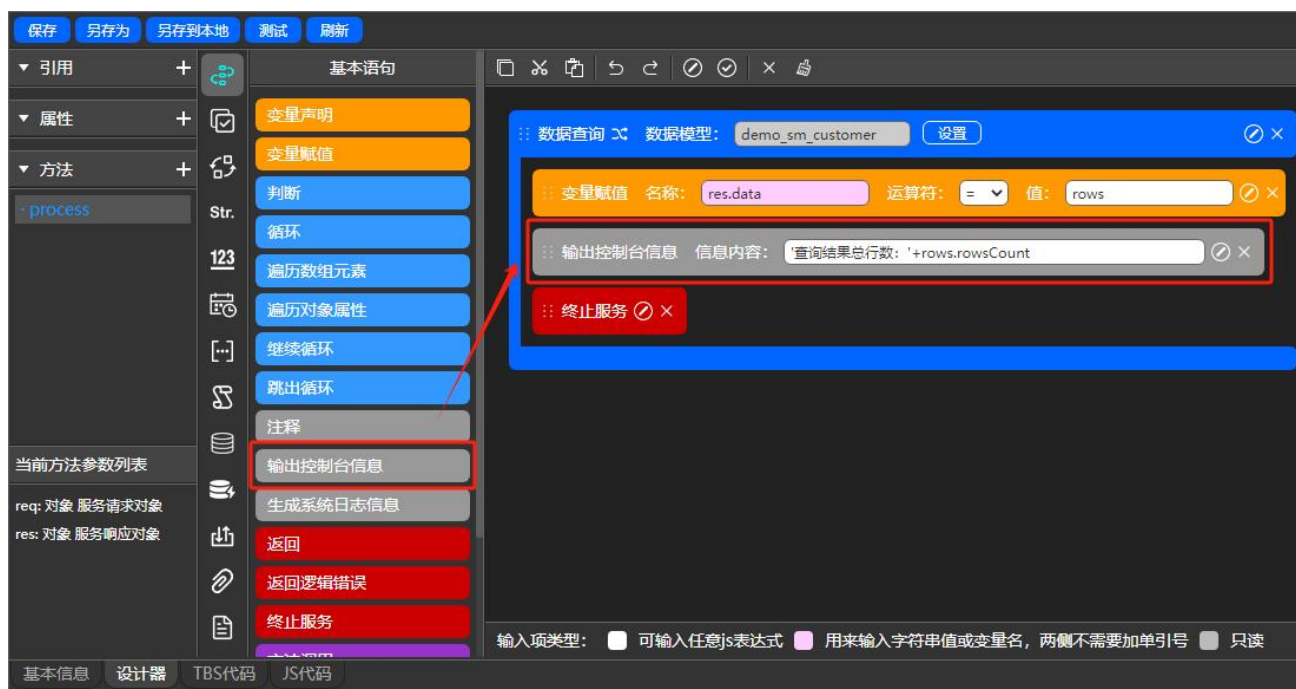
### 3.5.2.4 在后台服务内手动输出调试信息

前面介绍的日志信息基本都是任擎服务器自动生成的，如果我们要在自己的后台服务内手动输出一些调试信息，可以按照以下步骤操作：

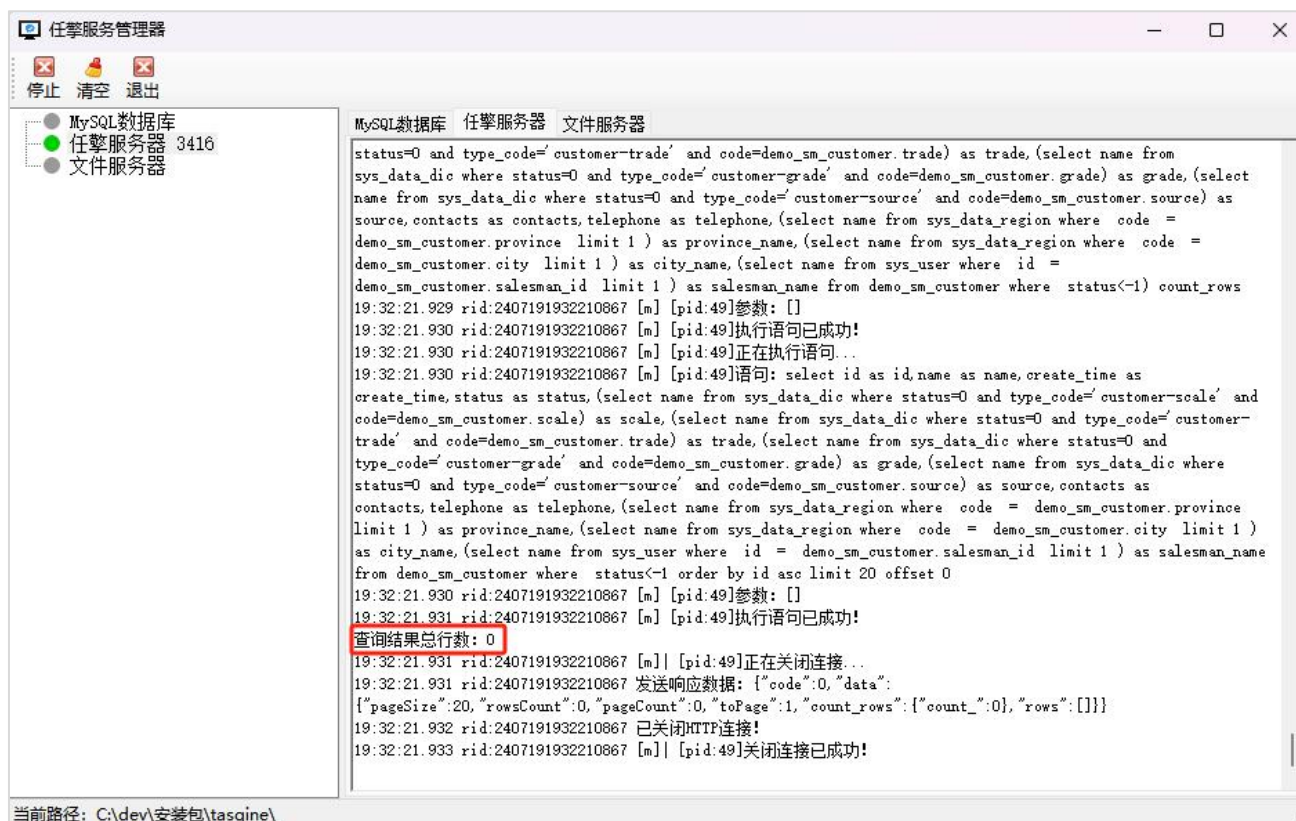
在销售管理示例项目的后台服务里，打开 `customer` 目录下的 `index_grid1_query.tbs` 这个后台服务文件，然后从基本语句里拖一条“输出控制台信息”语句到“终止服务”语句上面，该语句可以在任擎服务管理器或控制台中输出调试信息，但不会保存到日志文件里。里面的内容可以是任意 JavaScript 值、表达式或变量名，在这个示例页面里，我们将该语句的内容设置为：

`'查询结果总行数: '+rows.rowsCount`

设置完后的界面如下图所示：



然后点击工具栏上的“保存”按钮，保存该服务，然后再点击工具栏上的“测试”按钮，打开“后台服务模拟请求工具”，点工具栏上的“发送请求”按钮，可以模拟客户端请求该服务，然后在任擎服务管理器内就可以看到该语句的输出结果，如下图所示：



由此可以看到，我们在自己的后台服务里，随意可以输出一些信息以便进行调试。

### 3.5.2.5 配置日志相关参数

为了方便开发调试，任擎除了会将日志信息自动保存到日志文件中，而且默认还会将所有日志信息输出到任擎服务管理器的日志查看框或者 Linux 系统的终端控制台内，如果系统访问量比较大，频繁的日志输出会造成系统卡顿，所以，建议在正式环境内，禁用日志输出到控制台的功能，设置方法如下：

用文本编辑器打开 `tasgine\config\Tasgine.json` 文，该文件是任擎服务器的配置文件，里面保存的都是任擎服务器运行所需要的各种参数，包括网络端口、数据库连接参数、缓存加载选项等，其中有一个名为“Logger”的配置项，是用来配置与日志有关的参数的，将里面的 `outputToConsole` 参数设置为 `false`，重启任擎，日志信息就不会再输出到控制台中，但依然会保存到日志文件里，如果要查看日志信息，可以去 `tasgine\logs` 目录内查看。另外，“Logger”配置项里还有以下常用参数可以根据需要进行配置：

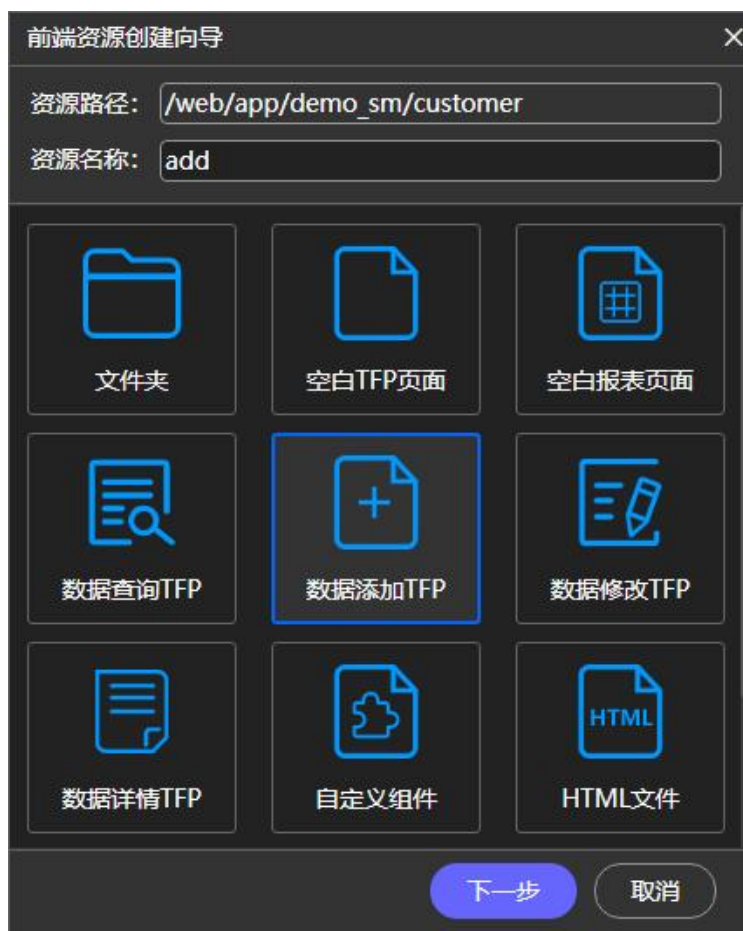
- **enabled**：是否启用日志跟踪器，默认为 `true`，正常情况下建议开启。
- **logFilePath**：日志文件保存位置，默认为 `/logs`，表示任擎服务器根目录下的 `logs` 子目录，可以使用绝对路径，例如：`D:\logs`。
- **logFileSaveDays**：日志文件保留时长，单位为天，默认为 30 天，超过保留天数会自动清除。
- **clearTime**：每日定期清理日志的时间，格式为 `HH:mm`，默认为：`00:30`。
- **logRequestHeader**：是否记录客户端请求的 HTTP 头信息，里面包含客户操作系统及浏览器类型和版本等信息，默认为 `false`。
- **logRequestData**：是否记录客户端请求的详细信息，默认为 `true`，建议开启。
- **logResponseData**：是否记录后台服务的响应信息，默认为 `true`，建议开启。
- **saveToDB**：可以通过该选项里面的子选项设置是否将相关日志信息保存到数据库中，例如动态服务或静态文件请求日志、文件上传下载日志等，以便统计分析，可以根据需要设置。



## 3.6 创建客户信息添加页面

### 3.6.1 选择页面类型

点击示例项目前端文件里 customer 文件夹右侧的加号，打开“前端资源创建向导”对话框，选中“数据添加 TFP”，资源名称会自动设置为 add，这里我们不用改，使用默认的即可。



然后点击下一步，可以打开“创建数据添加 TFP”向导对话框。

### 3.6.2 设置页面参数

在“创建数据添加 TFP”对话框中，可以对页面的基本信息和输入项进行设置，首先，还是要先选择前面创建的客户信息的数据模型，并设置页面标题为：添加客户信息，其他有些页面参数已经在上面介绍“创建数据查询 TFP”时已经介绍过了，这里就不再复述，这里重点讲

一下创建添加页面所需的相关参数，主要包括以下两个：

- **输入项文本标签显示位置：**用来设置在数据添加页面中，各个输入项的文本标签的显示位置，可以选择显示在输入项上部或左侧，默认是显示在输入项上部；
- **每行显示输入项数量：**用来设置在数据添加页面中，每一行显示的输入项数量，如果页面类型是对话框，则每行显示的输入项数量默认为 2，否则为 3。

然后，在下面的选项卡中，可以分别设置在前端页面里输入的字段信息和在后台服务里设置的字段。默认显示前端页面设置字段列表，在该列表中，点击字段前面的复选框，选择需要作为输入项显示在页面里的字段，然后可以设置这些字段的详细参数，包括：

- **输入项名称：**用来设置在数据添加页面里，显示在输入项上面或前面的文本标签的内容，默认等于字段的备注，另外，如果输入项不满足相关条件时，弹出的提示信息里也会显示该输入项的名称，而不是字段名；
- **输入项类型：**用来设置在数据添加页面里，用来输入字段值的输入项的类型，默认为单行文本输入框，如果要关联数据字典，则创建的页面里，会自动设置为数据字典组件；如果选择的编码规则，则需要在可选值里设置规则代码；注意：有些输入项类型不支持手机端和微信小程序，在创建手机端和小程序页面时，需要选择适合的输入项。
- **列数：**用来设置该字段对应的输入项在一行里占用几个输入项的宽度，如果一行显示 3 个输入项，则每个输入项（包括文本标签）占用的宽度，默认就是整个页面的  $1/3$ ，以此类推，当每行显示 2 个或 4 个输入项时，每个输入项的宽度默认就是整个页面宽度的  $1/2$  和  $1/4$ ，如果某个字段的内容比较多，则可以单独设置该输入项可以占用更多个输入项的宽度，但最多不能超过上面设置的每行显示的输入项数量；
- **必填：**用来设置该字段对应的输入项是否是必填，如果设置为必填，则在提交表单时，如果该字段没有设置值，则会收到提示，该输入项不能为空；

- **可选值**：如果输入项类型是单选框、多选框或者下拉列表，则可以设置可选值；
- **数据字典**：如果输入项的可选值需要关联数据字典查询，则可以在此输入框中双击鼠标，打开“选择数据字典”对话框，选择对应的数据字典；
- **默认值**：用来设置该字段在添加页面里显示的默认值；

另外，还可以点击字段右侧的上移、下移按钮调整输入项的显示顺序，点击删除按钮可以删除不需要的字段，对于添加客户信息的页面来说，需要在页面里显示的各个字段如下所示：

创建数据添加TFP

保存路径：

/web/app/demo\_sm/customer/add.tfp

数据模型：

demo\_sm/model/demo\_sm\_customer.tdm

选择...

页面标题：

添加客户信息

页面类型：

抽屉页

窗口大小：

宽

高

终端类型：

电脑端

背景颜色：

浅色

定位方式：

浮动定位

输入项文本标签显示位置：

☒ 输入项上部

☐ 输入项左侧

每行显示输入项数量：

3

仅支持PC

数据添加服务：

☒ 自动创建

☐ 使用已有服务

选择...

前端页面设置字段

后台服务设置字段

| <input checked="" type="checkbox"/> | 字段          | 输入项名称 | 输入项类型 | 列数 | 必填                                  | 可选值 | 数据字典(双击选择)      | 默认值 |
|-------------------------------------|-------------|-------|-------|----|-------------------------------------|-----|-----------------|-----|
| <input checked="" type="checkbox"/> | code        | 编码    | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> | name        | 名称    | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> | status      | 状态    | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-status |     |
| <input checked="" type="checkbox"/> | salesman_id | 销售员编号 | 用户选择  | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> | contacts    | 联系人   | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> | telephone   | 联系电话  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> | scale       | 客户规模  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-scale  |     |
| <input checked="" type="checkbox"/> | grade       | 客户等级  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-grade  |     |
| <input checked="" type="checkbox"/> | source      | 客户来源  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-source |     |

确定

取消

创建数据添加TFP

保存路径: /web/app/demo\_sm/customer/add.tfp 数据模型: demo\_sm/model/demo\_sm\_customer.tdm 选择...

页面标题: 添加客户信息 页面类型: 抽屉页 窗口大小: 宽 高

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧 每行显示输入项数量: 3 仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务 选择...

前端页面设置字段 后台服务设置字段

| 字段                                            | 输入项名称 | 输入项类型 | 列数 | 必填                                  | 可选值 | 数据字典(双击选择)      | 默认值 |
|-----------------------------------------------|-------|-------|----|-------------------------------------|-----|-----------------|-----|
| <input checked="" type="checkbox"/> telephone | 联系电话  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> scale     | 客户规模  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-scale  |     |
| <input checked="" type="checkbox"/> grade     | 客户等级  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-grade  |     |
| <input checked="" type="checkbox"/> source    | 客户来源  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-source |     |
| <input checked="" type="checkbox"/> trade     | 所属行业  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     | customer-trade  |     |
| <input checked="" type="checkbox"/> site      | 客户网址  | 单行输入框 | 2  | <input type="checkbox"/>            |     |                 |     |
| <input checked="" type="checkbox"/> province  | 所在省市  | 地区选择  | 1  | <input checked="" type="checkbox"/> |     |                 |     |
| <input checked="" type="checkbox"/> address   | 详细地址  | 单行输入框 | 2  | <input type="checkbox"/>            |     |                 |     |
| <input checked="" type="checkbox"/> comment   | 备注    | 单行输入框 | 3  | <input type="checkbox"/>            |     |                 |     |

确定 取消

从上面两图可以看到，在添加页面里，不需要显示所有字段，因为有些字段是在后台服务里设置的，例如创建时间、创建人编号等，不需要在添加页面里让用户输入，有些字段的数据不是在添加时生成的，例如删除人编号和删除时间是在进行删除操作时生成的。其中，有以下字段做了一些特殊设置：

- **status**：使用数据字典，字典编码设置为：customer-status；
- **salesman\_id**：输入项名称设置为所属销售，输入项类型使用用户选择组件；
- **scale**：使用数据字典，字典编码设置为：customer-scale；
- **grade**：使用数据字典，字典编码设置为：customer-grade；
- **source**：使用数据字典，字典编码设置为：customer-source；
- **trade**：使用数据字典，字典编码设置为：customer-trade；
- **site**：因为网站可能比较长，所以占用列数设置为 2；
- **province**：输入项使用地区选择组件，该组件可以级联选择省和市，所以 city 字段就不需

要再设置单独的输入项了；

- **address**：因为详细地址可能比较长，所以占用列数设置为 2；
- **comment**：输入项使用多行文本输入框，列数设置为 3；

然后点击“后台服务设置字段”选项卡，在此可以设置在后台服务中赋值的字段，这些字段不需要在前端页面里让用户手动输入，或者前端获取到的值可能不准确，例如系统时间，必须以服务器电脑的时间为准，因为用户电脑上的时间是可以随意修改的。

添加后台服务设置字段的方法如下：

先在字段列表里选择要添加的字段，然后在右侧选择系统宏变量，如果字段值不是系统宏变量，也可以不选择，然后点击最右侧的加号，即可完成添加。如果添加时没有选择系统宏变量，则可以在列表里直接设置字段的值，这里设置的值必须是有效的 JavaScript 表达式。

创建数据添加TFP

保存路径: /web/app/demo\_sm/customer/add.tfp 数据模型: demo\_sm/model/demo\_sm\_customer.tdm 选择...

页面标题: 添加客户信息 页面类型: 抽屉页 窗口大小: 宽 高

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧 每行显示输入项数量: 3 仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务 选择...

前端页面设置字段 后台服务设置字段

字段列表: 系统宏变量: +

| 字段名         | 字段值(JS表达式)         |
|-------------|--------------------|
| creator_id  | req.session.userId |
| create_time | new Date()         |

确定 取消

对于添加客户信息来说，以下几个字段需要在后台服务里设置：

- **creator\_id**：创建人编号，需要设置为当前用户编号：req.session.userId；

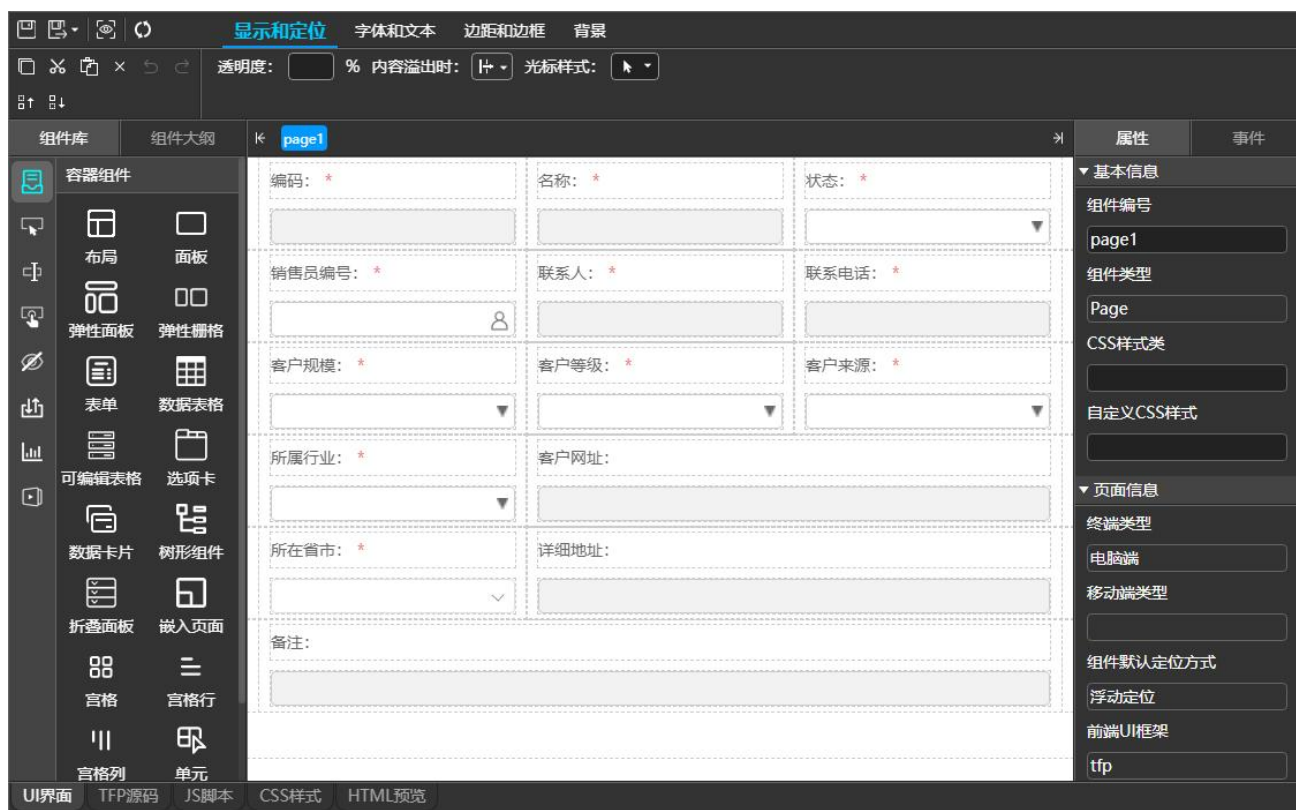


- **create\_time**: 创建时间, 需要设置为当前时间: `new Date()`;

都设置好之后, 点击确定按钮, 即可完成客户信息添加页面的创建。

### 3.6.3 查看客户信息添加页面

用向导创建完添加页面后, TaskBuilder 会自动用前端页面设计器打开, 界面如下图所示:



在此可以继续修改完善该页面, 选择各个组件, 可以设置其样式、属性和事件, 可以删除不要的组件, 可以从左侧组件库中拖拽新的组件。

其中, 有几个特殊的组件需要单独说明一下:

- **用户选择组件**: 该组件用来从组织结构中选择指定的人员, 提交表单时, 该组件默认传给后台服务的是选择的人员都编号 (id), 如果要传其他信息, 可以在该组件的属性里设置“值类型”, 可以选择: 用户编号 (id)、用户代码 (code)、用户姓名 (name)、用户对象 (包含前面三种信息的 JSON 对象); 如果在该组件的属性里选中了“默认为当前用户”, 则打开页面时, 该组件会自动显示当前用户的信息;



- **数据字典组件：**该组件用来选择在后台管理系统的系统维护的数据字典模块中维护的指定类型的字典信息，选中该类型组件后，点击属性里的“字典类别编码”，可以打开选择数据字典类别的对话框，在此选择对应的字典类别，点确定即可；
- **地区选择组件：**该组件用来级联选择省市等地区信息，目前收录的数据是截止到 2023 年的我国所有省（直辖市、自治区）、市、区县、乡镇街道四级行政区域的数据，默认仅显示省和市两级数据供选择，如果要显示更多级别，可以在属性里设置地区类别，支持：省市、省市县、市县、市县乡、县乡，如果是后面三种类型，需要在属性里设置“顶层地区代码”，例如，如果地区类别设置为市县，则需要将“顶层地区代码”设置为具体省的地区代码，如果地区类别设置为县乡，则需要将“顶层地区代码”设置为具体市的地区代码，目前最多仅支持 3 级级联选择，不支持 4 级，因为 4 级的数据量太大，会造成系统卡顿。另外，该组件传给后台的值为地区代码组合，例如如果地区类别设置为省市，则传给后台服务的值为：省级代码/市级代码，如果在数据库中需要将各级代码分别存到多个字段中，则需要进行拆分，具体方法见下面的描述。

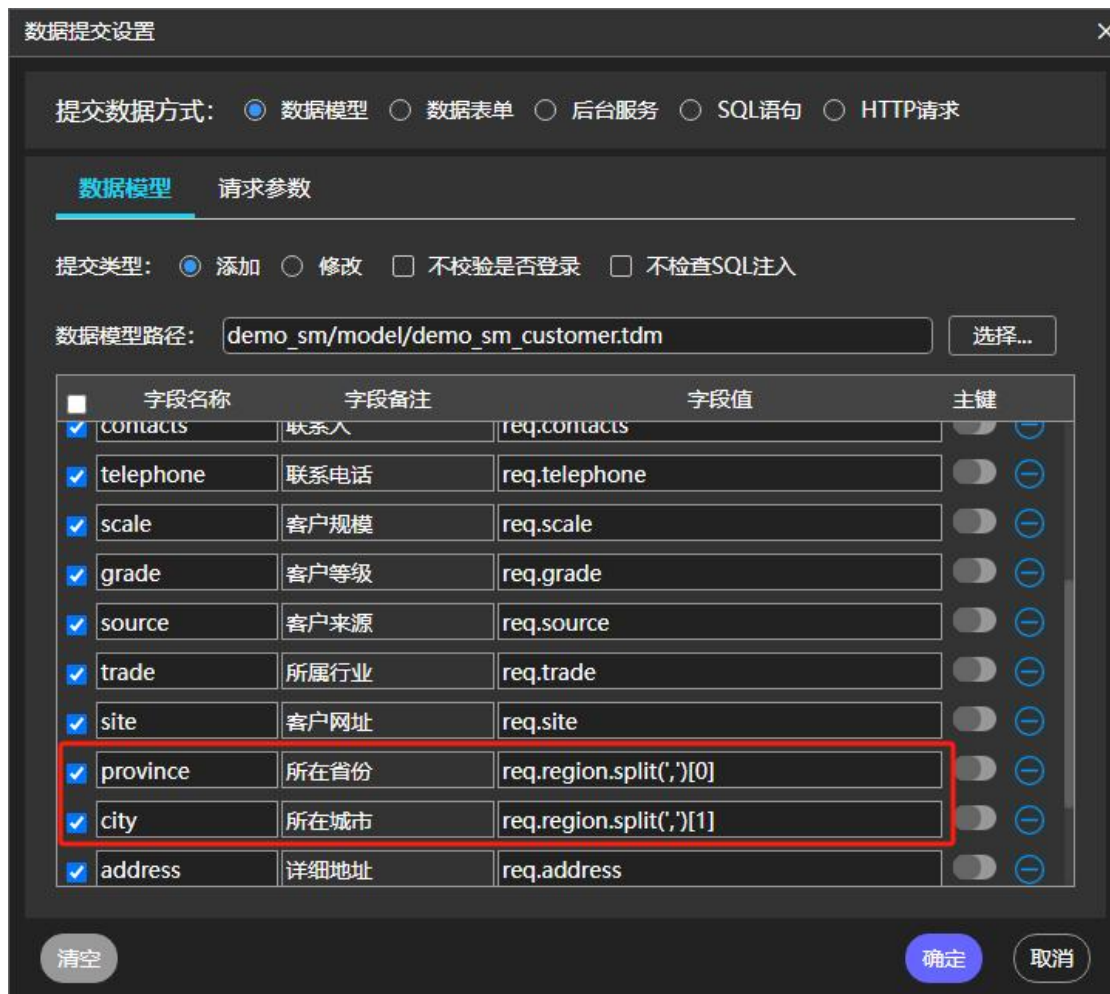
### 3.6.4 修改所属省市字段设置

在前面用向导创建页面时，所在省市输入项使用的是地区选择组件，该组件最终发给后台服务的值是“省级代码/市级代码”这种格式，而我们设计的客户信息数据模型里，省和市是分两个字段存储的，这种情况在向导里没办法处理，需要在创建完页面后，在前端页面设计器里设置，具体的设置方法如下：

(1) 首先，在页面里选中所在省市对应的地区选择组件，在属性设置里，将该组件的编号改成：region，如下图所示：



(2) 然后，在页面里选中表单组件 form1，点击属性设置里面的“数据提交设置”，打开如下图所示对话框：



在该对话框中，将所在省份和所在城市分别设置为以下值：

- **province:** req.region.split('/')[0] //表示从前端页面所在省市输入项的值里取/前面的编码
- **city:** req.region.split('/')[1] //表示从前端页面所在省市输入项的值里取/后面的编码

上面的 JavaScript 脚本里的 split()函数的作用是将字符串按指定的分隔符拆分为字符串数组，然后 [] 的作用是取数组里的第几个元素，元素索引从 0 开始，例如上面的所在省市输入项传给后台的值为：44/4403，其中 44 是广东省的地区编码，4403 是深圳市的地区编码，则

用：req.region.split('/')[0] 得到的值就是 44，req.region.split('/')[1]得到的值就是 4403。

设置好之后，点确定按钮关闭对话框，然后点击工具栏上的保存按钮保存当前页面的修改，这时候，就会自动重新生成添加客户信息的后台服务，然后就可以预览测试了。

### 3.6.5 测试客户信息添加页

现在我们用预览工具测试一下客户信息添加页面是否能正确运行。

因为客户信息添加页面是抽屉页，而且添加完数据要在列表页面里显示，所以，需要从列表页面里打开该页面，不建议直接预览添加页面进行测试。

在 TaskBuilder 里打开前面创建的客户信息列表页面，然后点击工具栏上的预览按钮，会在预览工具里打开客户信息列表页面，点击该页面上部面板里的添加按钮，即可以看到，从界面右侧会滑出一个抽屉页面，里面显示的内容就是我们创建的客户信息添加页面，从该页面可以看到，每行默认显示三个输入项，每个输入项的文本标签显示在输入项上部，必填输入项的文本标签右侧有红色的 \* 号提示，所示销售输入项为用户选择组件，客户规模、所属行业、客户等级、客户来源这几个输入项为数据字典，点击后可以在下拉列表里选择，所在省市为省市两级级联选择组件，可以同时选择省份和城市，备注输入项为多行文本，可以输入比较多的文字内容，其他输入项都是默认的单行文本。设置完客户信息后，整体界面如下图所示：

在该抽屉页中输入客户有关的信息，然后点击右上角的确定按钮，即可实现客户信息的添加，并且会自动关闭抽屉页，然后刷新列表页面里的数据表格组件，重新加载数据，也就能看到我们刚才添加的那条客户信息，界面如下：

刷新

本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

查询条件

客户名称:

联系人:

客户规模:

所属行业:

客户等级:

客户来源:

所属销售:

状态:

查询

添加

| 名称              | 创建时间       | 销售员 | 联系人 | 联系电话        | 客户规模 | 所属行业 | 客户等级 | 客户来源 |
|-----------------|------------|-----|-----|-------------|------|------|------|------|
| 深圳某电气设备生产制造有限公司 | 2024-07-20 | 杨花花 | 李总  | 13678905432 | 小微   | 生产制造 | A级   | 电话咨询 |

<< < 1 > >> 到第 1 页

共 1 页, 1 条记录 20

## 3.7 抽屉页使用简介

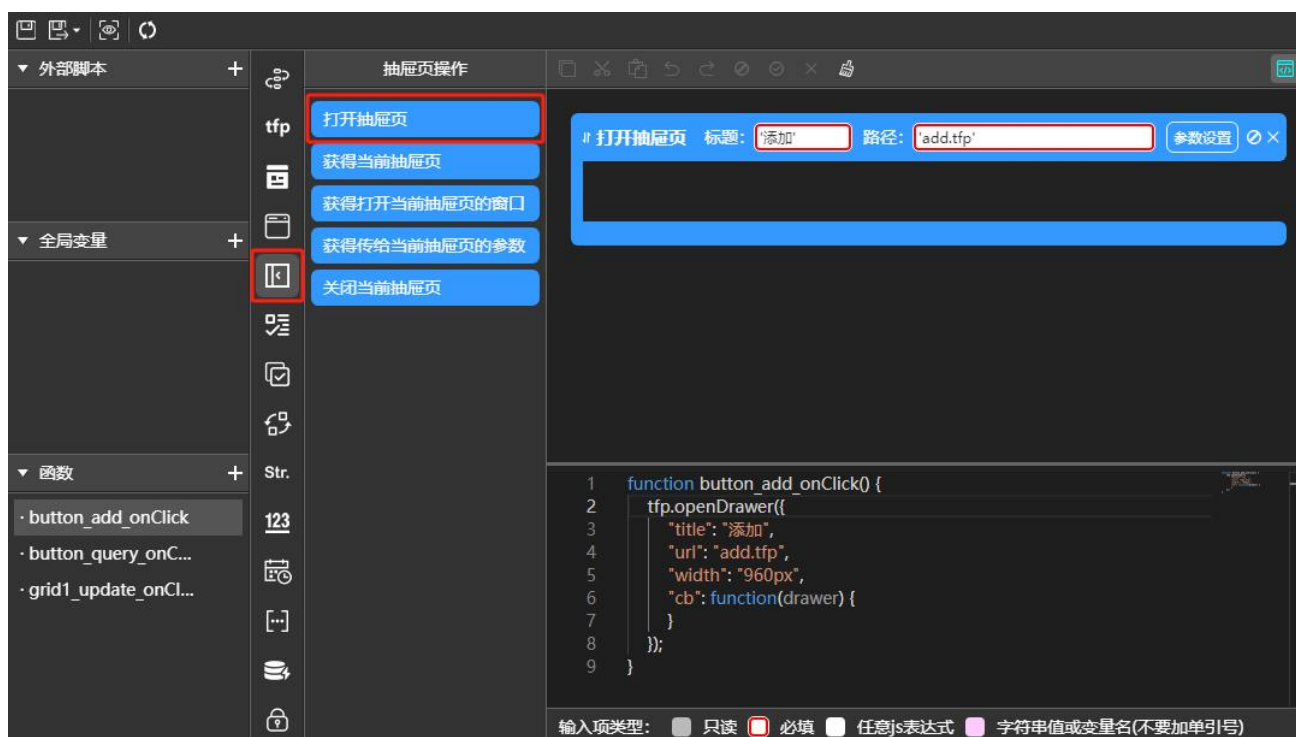
前面介绍的客户信息添加页面是采用的抽屉页形式显示的，为了让大家能更深入地了解抽屉页的使用，下面介绍一下抽屉页有关的知识。

### 3.7.1 抽屉页的定义

一个页面是否是抽屉页是由打开方式决定的，如果用 `tfp.openDrawer()` 方法打开，就会以抽屉页形式展示，这个页面也就是我们所说的抽屉页，如果用 `tfp.openPage()` 打开，则就是普通页面。在设计时，抽屉页与普通页面并没有什么区别，只有一点，就是如果抽屉页需要显示确定按钮，并希望点击该按钮后执行一些操作，则需要设置页面（page1）组件的“点击抽屉页确定按钮时”事件，在里面编写相应脚本，否则点击确定按钮后会没有任何反应。

### 3.7.2 打开抽屉页

在 TaskBuilder 内打开客户信息列表页面，选中“添加”按钮，在右侧的事件设置里，双击“点击时”事件输入框，可以看到如下图所示的脚本设置界面，里面放置了一个“打开抽屉页”的脚本语句，该语句是从左侧语句库里的“抽屉页操作”中拖拽过来的，在这条语句的设置面板里，可以直接输入抽屉页的标题和路径，如下图所示：



**注意：**用前端资源创建向导创建列表页面时，如果选择了显示添加按钮，则添加按钮里打开的添加页面的文件名默认为 add.tfp，如果我们后面创建的添加页面不是这个文件名，则需要在此进行修改，否则，预览时，点击添加按钮，会出现“请求地址无效”的错误：



另外，在也可以点击该语句到“参数设置”按钮，设置更详细的参数，界面如下：

参数设置

语句类型: 打开抽屉页

| 参数名       | 必填 | 参数类型 | 参数值       |
|-----------|----|------|-----------|
| 标题        | 是  | 表达式  | '添加'      |
| 路径        | 是  | 表达式  | 'add.tfp' |
| 宽度        | 是  | 表达式  | '960px'   |
| 确定按钮文字    | 是  | 表达式  | '确定'      |
| 隐藏确定按钮    | 否  | 布尔值  |           |
| 点击其他区域时关闭 | 否  | 布尔值  |           |
| 传入参数      | 否  | 变量名称 |           |
| 回调参数名     | 否  | 字符串值 | drawer    |

如果参数类型是表达式, 则可输入任意js表达式, 这时如果输入字符串, 两边需要加单引号; 如果参数类型是字符串值, 则直接输入字符串即可, 两边不需要加单引号。

确定

取消

在此除了标题和路径, 还可以设置以下参数:

- **宽度:** 用来设置抽屉页的宽度, 默认为 960px, 一般情况使用默认的即可, 不建议设置得太宽或太窄, 抽屉页的高度不需要设置, 会自动与浏览器内容区的高度保持一致;
- **确定按钮文字:** 默认为“确定”这两个字, 可以改成其他文字, 但不建议太长;
- **隐藏确定按钮:** 可以设置是否隐藏确定按钮, 默认是不隐藏, 如果抽屉页里的内容仅用于查看, 不需要录入数据并提交到后台, 则可以隐藏确定按钮, 例如数据添加或修改页面;
- **点击其他区域时关闭:** 打开抽屉页后, 默认需要点击确定或取消按钮才能关闭, 但对于仅显示查询信息的页面, 可以将该参数设置为 true, 然后打开抽屉页后, 点击浏览器里抽屉页以外的区域时就会自动关闭抽屉页, 不需要点确定或取消按钮;
- **传入参数:** 如果打开抽屉页时, 想给抽屉页传递一些数据, 可以将这些数据保存到一个变量内, 然后在这里设置这个变量的名称, 则在打开抽屉页后, 就可以在抽屉页的页面加载



时事件里，使用前端脚本语句库的“抽屉页操作”里的“获得传给当前抽屉页的参数”语句获得该参数的值；

- **回调参数名：**打开抽屉页后，会执行一个回调函数，在该回调函数里，会传入一个参数，对应当前抽屉页页面的 window 对象，该参数默认为名为 drawer，可以在此修改；通过该参数可以获得抽屉页里的各个组件，然后执行相关操作，例如在回调函数里调用抽屉页里表单组件的提交方法：`drawer.form1.submit()`；

### 3.7.3 设置抽屉页扩展按钮

除了确定和取消按钮以外，如果还想在抽屉页顶部显示其他按钮，需要使用脚本打开抽屉页，并设置相关选项，示例脚本如下：

```
tfp.openDrawer({
  "title": "添加",
  "url": "add.tfp",
  "width": "960px",
  "okButtonText": "确定",
  "exButtons": [{
    "text": "扩展按钮 1",
    "color": "#FFFFFF", //按钮文字颜色
    "bgColor": "#3366FF", //按钮背景颜色
    "onClick": function(drawer) {
      console.log("点击了扩展按钮 1");
    }
  }],
  "text": "扩展按钮 2",
  "color": "#FFFFFF", //按钮文字颜色
  "bgColor": "#666666", //按钮背景颜色
  "onClick": function(drawer) {
    console.log("点击了扩展按钮 2");
  }
});
```

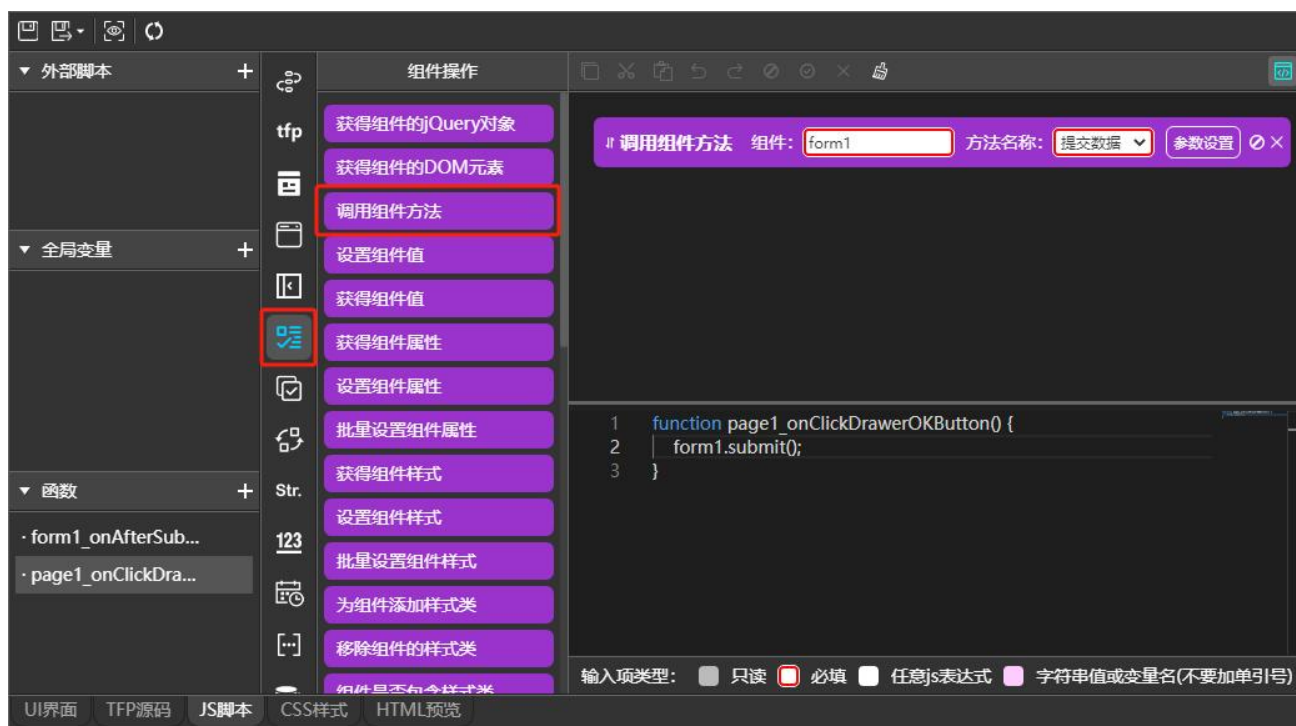
```
    }  
  },  
  "cb": function(drawer) {  
    console.log("点击了确定按钮");  
  }  
});
```

从上面的脚本可以看到，在用 `tfp.openDrawer()` 方法打开抽屉页时，可以通过 `exButtons` 参数设置一个或多个扩展按钮，每个按钮可以设置以下参数：

- **text**：用来设置扩展按钮的文字
- **color**：用来设置扩展按钮的文字颜色
- **bgColor**：用来设置扩展按钮的背景颜色
- **onClick**：用来设置点击扩展按钮之后执行的回调函数

### 3.7.4 设置抽屉页确定按钮事件

如果抽屉页是用来添加和修改数据的，则要保留右上角的“确定”按钮，并要设置该按钮的点击事件，在该事件内提交表单，具体操作如下：

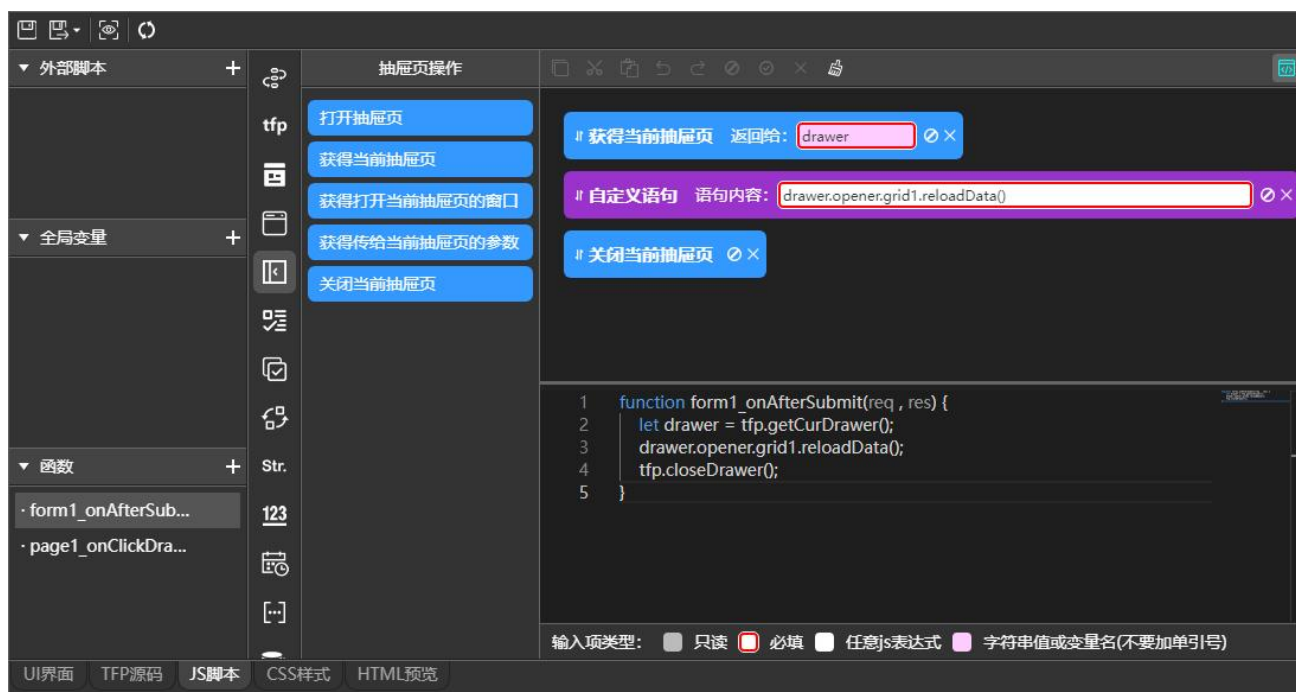


在 TaskBuilder 内打开前面创建的添加客户信息页面，选中页面组件（page1），然后双击右侧事件设置面板里的“点击抽屉页确定按钮时”事件输入框，打开如上图所示的脚本设计器。

从上图可知，在该事件处理函数中，从语句库的组件操作中拖拽了一条“调用组件方法”语句，并将组件设置为 form1，方法设置为提交数据（submit），这样设置后，在点击抽屉页的确定按钮时，就会将该页面表单里的数据提交给后台服务。

### 3.7.5 设置抽屉页内表单提交后事件

在客户信息添加页里选中表单组件（form1），然后双击右侧事件设置面板里的“提交后”事件输入框，打开如下图所示的脚本设计器：



从上图可知，在该事件处理函数中，执行了以下操作：

- (1) 首先，通过“获得当前抽屉页”语句获得当前抽屉页对象，赋值给变量 drawer；
- (2) 然后，通过该变量获得打开当前抽屉页的列表页面的 window 对象（drawer.opener），并通过这个对象调用列表页面里数据表格组件的重新加载数据的方法：grid1.reloadData()；
- (3) 最后用“关闭当前抽屉页”语句关闭当前抽屉页。

上面的脚本中，实际就是两步，第一步是在抽屉页的表单提交成功后，刷新列表页面里的数据表格，以便显示添加或修改的内容，这里最关键的就是要通过 `drawer.opener` 获得打开当前抽屉页的列表页面的 `window` 对象，通过该对象就可以获得列表页里的各个组件，并调用这些组件的方法，这里最常见的就是让数据表格组件重新加载；然后第二步就是关闭当前抽屉页。

## 3.8 完善客户信息列表页面

从上图可以看到，该列表页面存在一些问题，包括：

- 客户名称列太窄，内容出现了串行
- 客户状态显示的数字，不友好
- 客户规模、客户级别这种有固定可选值的列，可以根据不同的值显示不同的颜色

下面介绍一下怎么进行完善。

在 TaskBuilder 里打开客户信息列表页面，并选中数据表格组件 `grid1`，然后点击属性里的“数据列设置”，就会打开数据列设置对话框，然后按下面的介绍分别进行设置。

### 3.8.1 数据表格列设置简介

下图所示的数据表格组件的“数据列设置”对话框从左至右分为三个区域：

- **数据表格列：**在此列出了数据表格的所有列，点击某个列，中间的选项分类和右侧的选项列表的内容会随着改变；点击左下角的加号，可以添加新的列；
- **选项分类：**在界面中间部分显示的是当前选中的列可以设置的选项分类，其中，基本信息、排序设置、显示条件和行转列设置是所有列都有的类别，然后，不同类型的列，会有对应的选项分类，例如文本类型的列有文本设置，按钮类型的列有按钮设置等。
- **选项列表：**在中间点击某个选项分类，会在右边显示这个分类里的各个选项列表，并显示当前选中的列这些选项的值；

这个设置窗口参数很多，这里不展开介绍，详细信息请参考数据表格组件的 API 文档。

数据列设置

数据表格列

名称

创建时间

销售员

联系人

联系电话

客户规模

所属行业

客户等级

客户来源

所在省市

状态

修改

删除

选项分类

基本信息

文本设置

排序设置

显示条件

行转列设置

选项列表

列名

名称

内容类型

文本

水平对齐方式

左对齐

垂直对齐方式

顶端对齐

列宽度

300

背景颜色

位置固定

数据绑定表达式

{name}

点击后执行的脚本

CSS样式class

自定义CSS样式

确定

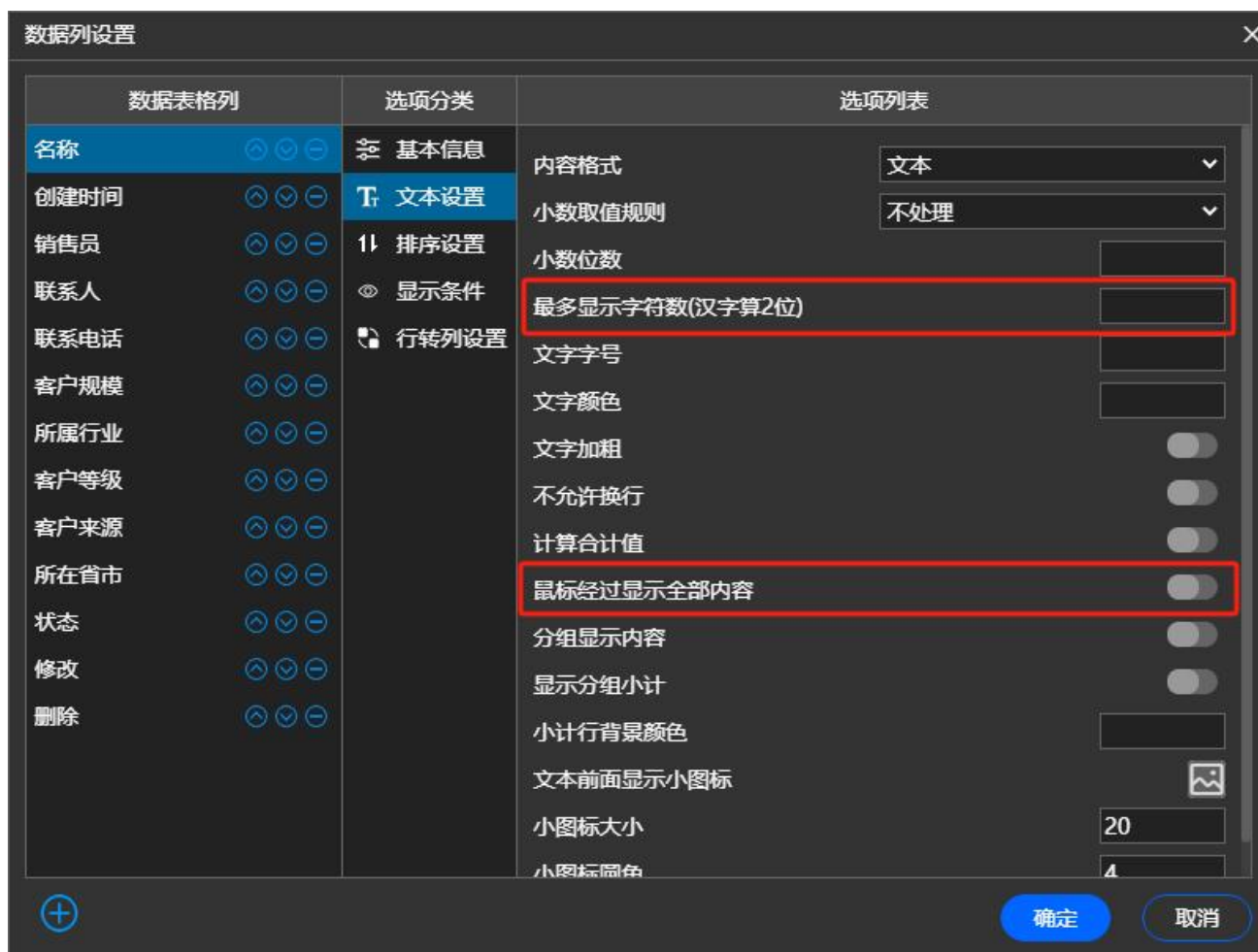
取消

### 3.8.2 客户名称列宽度调整

在上面的数据列设置对话框里，选中“名称”这一列，然后在选项分类里选中“基本信息”，然后在右侧的选项列表里，把“列宽度”设置为 300，然后点确定按钮，即可完成客户名称列宽度的设置，然后再点击工具栏上的“预览”按钮，这时可以看到“名称”这一列变宽了，里面的内容也不串行了。

| 名称              | 创建时间       | 销售员 | 联系人 | 联系电话        | 客户规模 | 所属行业 |
|-----------------|------------|-----|-----|-------------|------|------|
| 深圳某电气设备生产制造有限公司 | 2024-07-20 | 杨花花 | 李总  | 13678905432 | 小微   | 生产制造 |

其他字段的宽度也可以根据实际需求进行设置，设置的原则就是既不要造成内容串行，也不要内容两边出现很多空白，浪费空间。



数据列表里不建议显示内容非常多的字段，例如公告新闻的正文，都显示在列表里会造成查询速度缓慢，页面也不美观。如果确实需要在数据列表里显示一些摘要性质的内容，可以在该列的文本设置里设置以下两个参数来进行优化：

- **最多显示字符数**：在此可以设置该列内最多显示的字符数，超过会自动截取，汉字算两个字符；
- **鼠标经过显示全部内容**：启用该选项后，如果该列的内容被截取，当鼠标放在该列上时，就会自动弹出一个层显示该列的全部内容。

### 3.8.3 客户名称列增加小图标

到目前为止，我们开发完成的客户信息列表页面里，显示的都是纯文字，看着比较素，为了增加美观度，可以为客户名称这一列增加一个小图标，具体操作如下：



在数据表格组件的数据列设置对话框内，选中“名称”这一列，然后在选项分类里选中“文本设置”，然后在右边选项列表里，点击“文本前面显示小图标”选项右侧的图标按钮：



打开如下对话框：

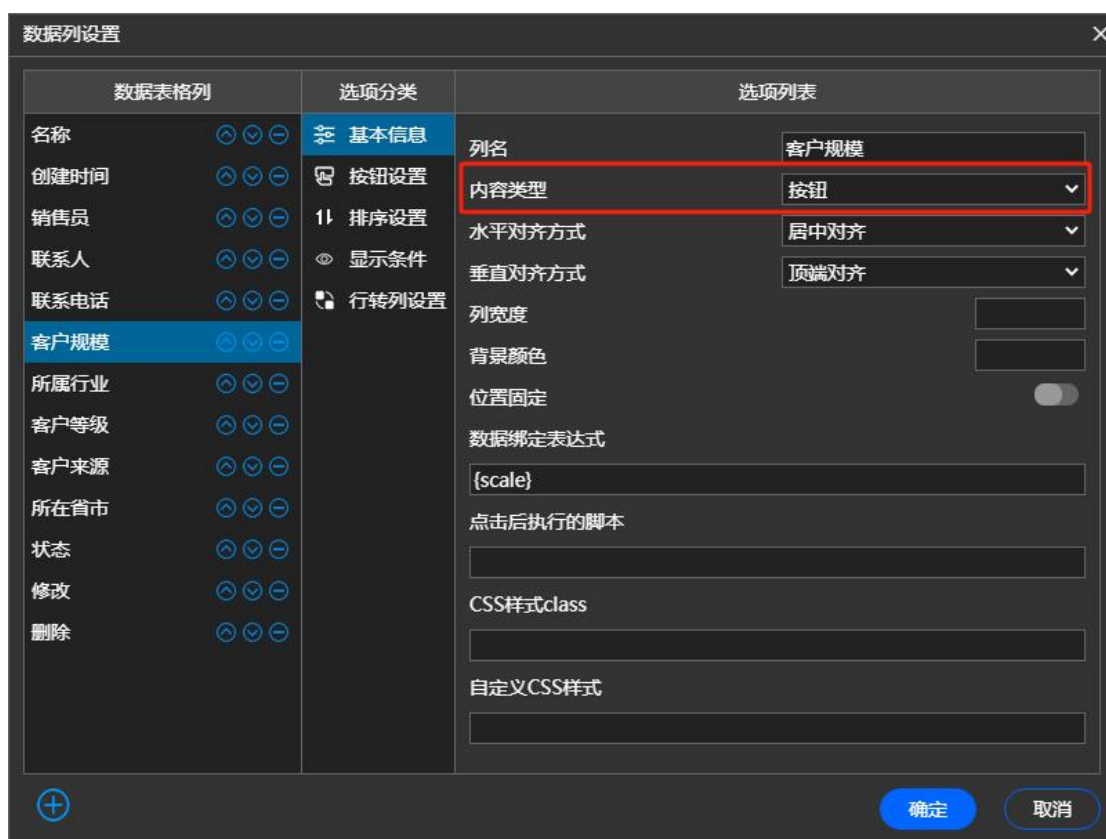


在此先在左边选择图标，然后在右边选择背景颜色，然后点保存按钮，然后再点击数据列设置对话框里的确定按钮，再保存列表页面，再预览，就可以看到如下所示效果：

| 名称                                                                                                | 创建时间       | 销售员 | 联系人 | 联系电话        | 客户规模 | 所属行业 |
|---------------------------------------------------------------------------------------------------|------------|-----|-----|-------------|------|------|
|  深圳某电气设备生产制造有限公司 | 2024-07-20 | 杨花花 | 李总  | 13678905432 | 小微   | 生产制造 |

### 3.8.4 客户规模列显示样式美化

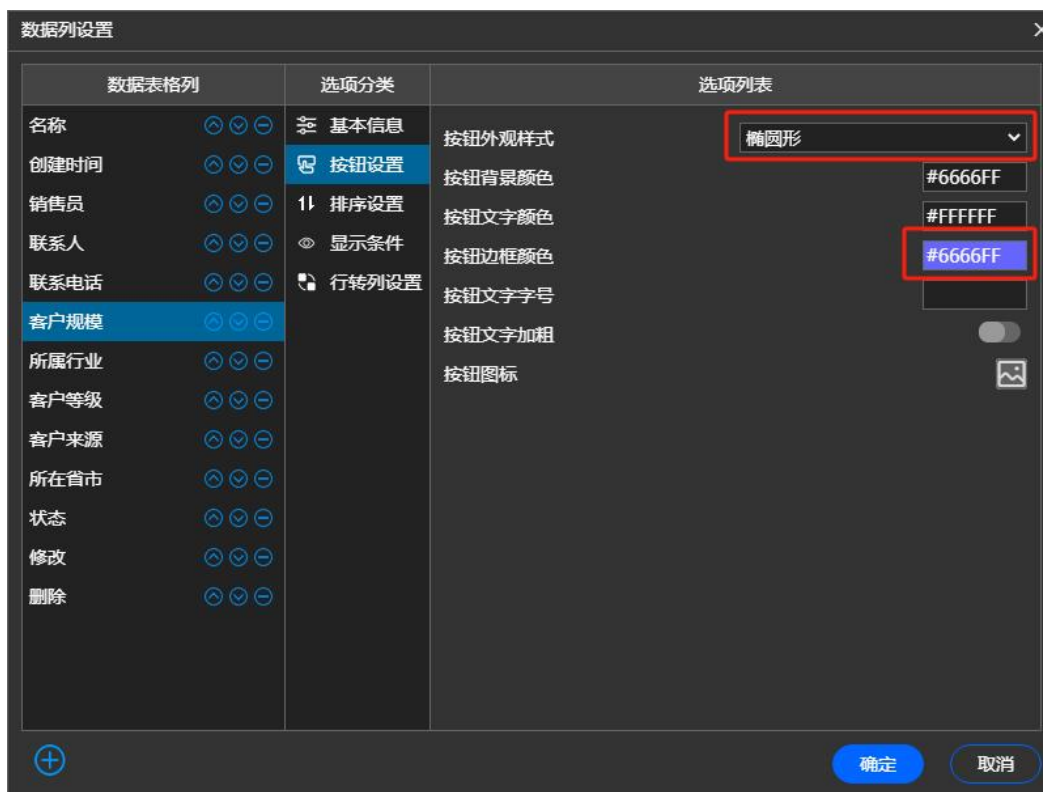
客户信息列表中的客户规模和客户等级这两列都是通过数据字典配置的，而且可选值是固定的几个值，为了让列表更加美观，可以将这两列的内容类型设置为按钮，然后再设置显示条件，根据不同的值显示不同的文字颜色和背景颜色。具体操作方法如下：



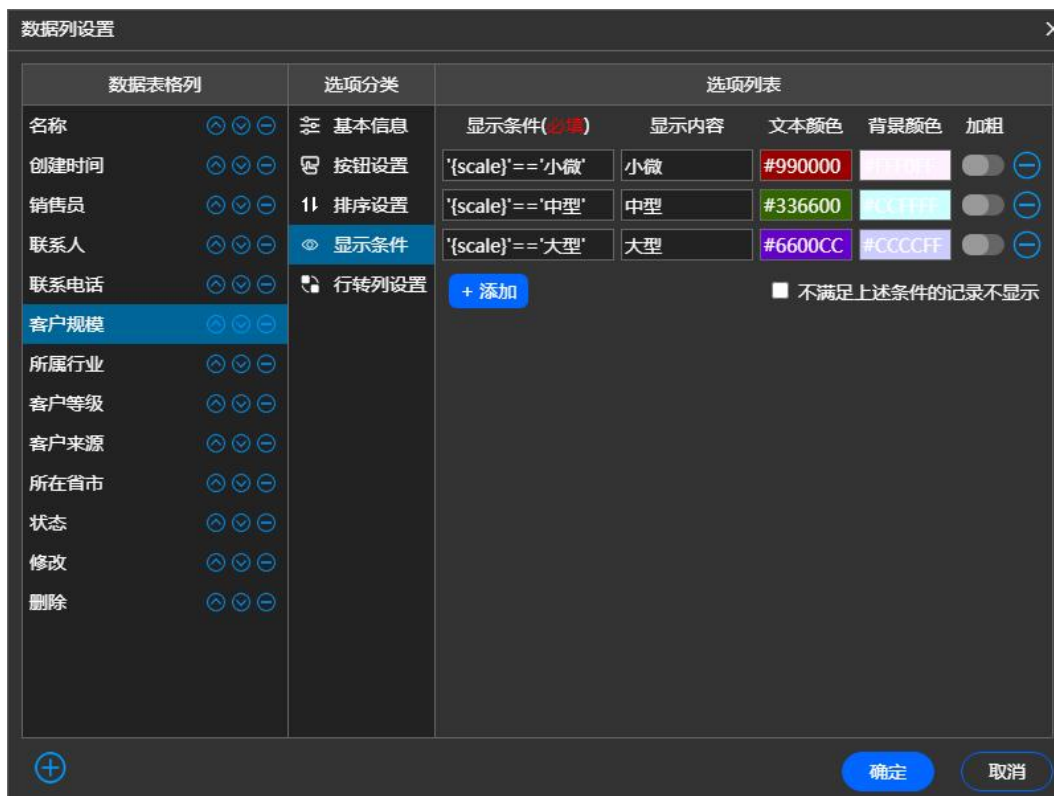
在数据表格的数据列设置对话框中，选择“客户规模”这一列，然后在基本信息的选项列表里，将内容类型改成“按钮”，如上图所示。

这时选项分类里的“文本设置”变成了“按钮设置”，点击“按钮设置”，将按钮外观样

式设置为“椭圆形”，并设置一下按钮边框颜色，如下图所示：



然后点击选项分类里的“显示条件”，在此可以添加多个条件，分别根据不同的客户规模显示对应的内容和文本及背景颜色，如下图所示：



在上图所示的显示条件设置界面里，可以点击蓝色的“+添加”按钮，添加显示条件设置行，点击每行后面的删除按钮可以删除该条件，在每行条件里可以设置如下信息：

- **显示条件：**在此设置具体的显示条件，必须是有效的 JavaScript 表达式，里面可以用 **{字段名}** 的形式嵌入数据表格里当前行数据的指定字段的值，例如上图中所示的：**{status}==1**，即表示如果客户状态编号等于 1，就显示后面显示内容里设置的“已签约”，并且文字颜色为后面的文本颜色输入框里设置的颜色。如果字段的数据是字符串类型，则需要在 {} 两边加上单引号，否则会报错，例如：**{grade}=='A 级'**，注意，在 JavaScript 语言中，判断是否等于的运算符是：**==**，即两个等于号，不是一个等于号，一个等于号是赋值运算符，判断是否不等于的运算符是：**!=**；
- **显示内容：**用来设置满足前面的显示条件后要显示的内容，一般是固定的字符，不需要加单引号；
- **文本颜色：**用来设置满足前面的显示条件后，显示内容的文本颜色，默认黑色；
- **背景颜色：**用来设置满足前面的显示条件后，该单元格的背景颜色，默认无背景；
- **加粗：**用来设置满足前面的显示条件后，显示内容是否加粗，默认不加粗；

这里的文本颜色和背景颜色不一定非得按上图中一模一样设置，选择您觉得看着舒服的颜色即可，但文字和背景的颜色不要都是深色或都是浅色，应该深浅搭配，否则文字会看不清。

在客户列表里再添加两条客户信息，客户规模分别设置为小型和大型，则最后界面如下：

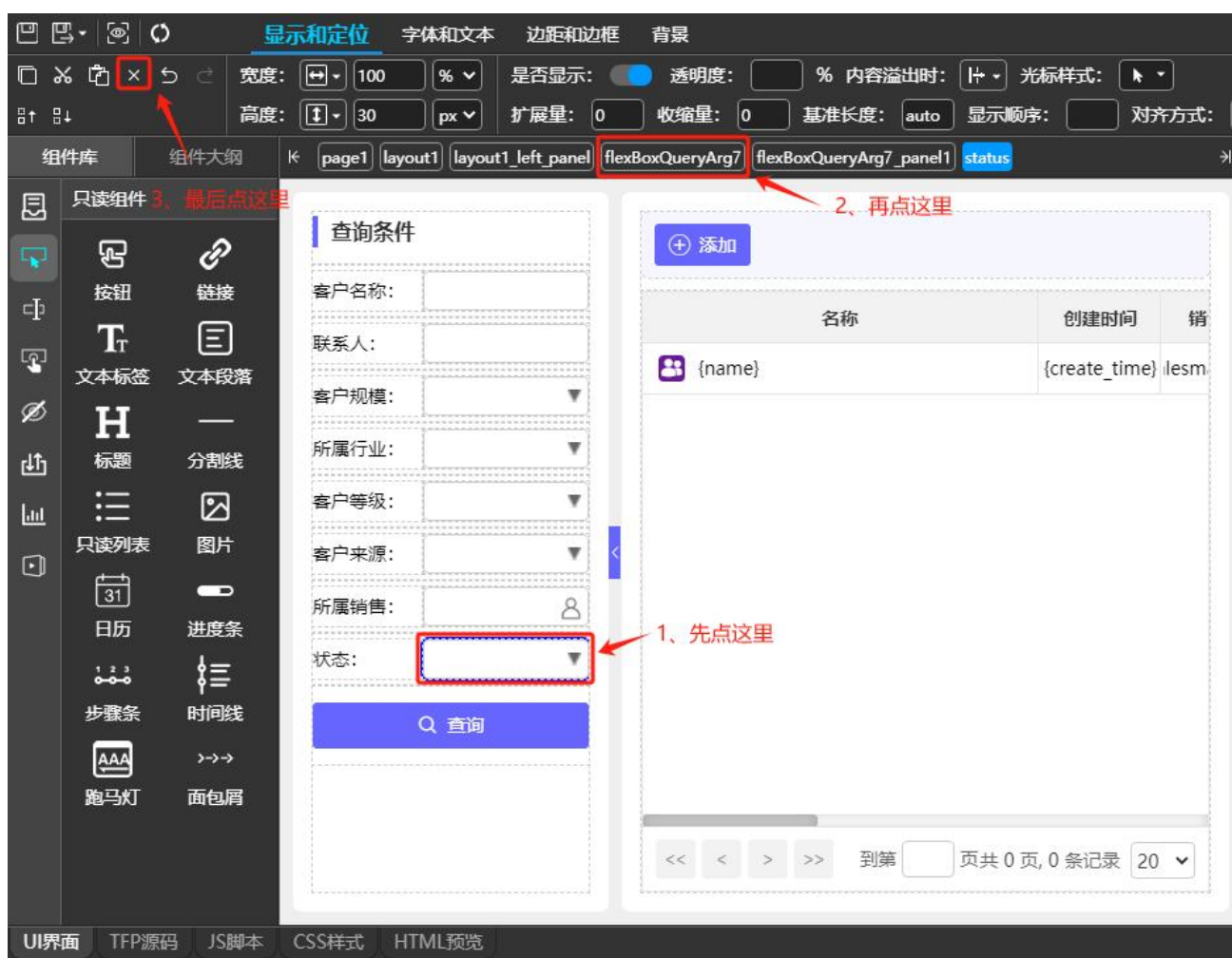
| 名称              | 创建时间       | 销售员 | 联系人 | 联系电话        | 客户规模 | 所属行业 | 客户等级 |
|-----------------|------------|-----|-----|-------------|------|------|------|
| 深圳某电气设备生产制造有限公司 | 2024-07-17 | 杨花花 | 李总  | 13677889900 | 中型   | 生产制造 | B级   |
| 东莞银汇消防工程有限公司    | 2024-07-18 | 孙研  | 张总  | 13799087654 | 小微   | 生产制造 | B级   |
| 佛山科越医疗器械有限公司    | 2024-07-18 | 刘鑫  | 马总  | 19232145678 | 大型   | 生产制造 | A级   |

从图中可以看到，不同规模的客户，在表格里以不同颜色的椭圆形显示，比纯文字更加美观，其他列，包括客户等级、客户状态等也可以参考该列进行美化，具体操作就不再复述。

### 3.8.5 修改列表页数据查询设置

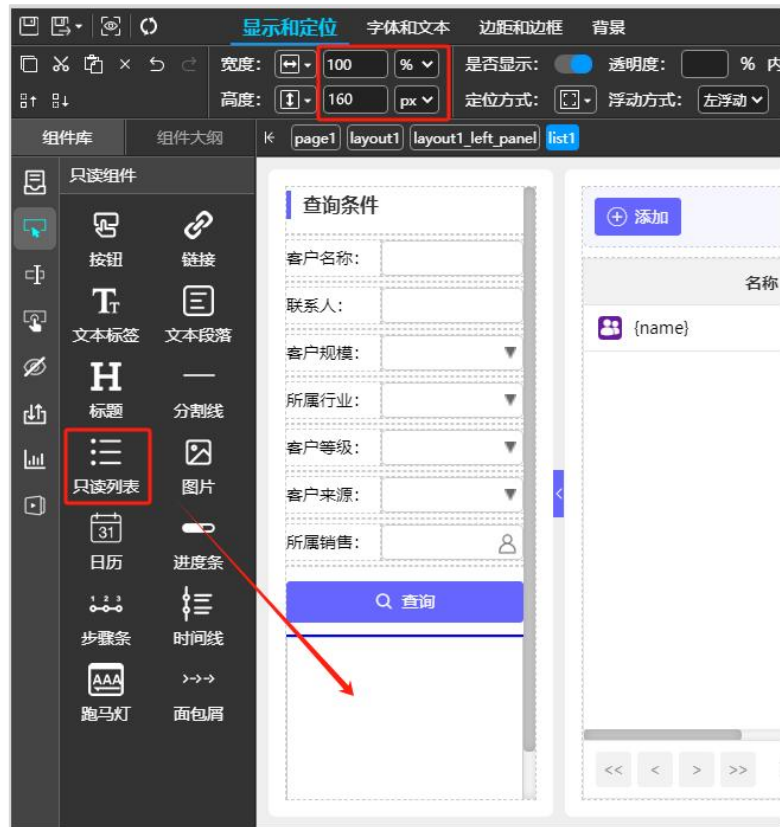
在前面创建的客户信息列表页面里，左侧查询条件里的客户规模、客户等级、客户状态等输入项的可选项是固定的几个值，可以更换成“只读列表”组件，直接把可选值显示出来供用户选择，这样会更加直观，现在以客户状态为例，教大家怎么实现。

首先选中客户状态查询条件下拉框，然后在上部组件路径栏里点击这一行的弹性栅格组件：flexBoxQueryArg9，然后点工具栏上的删除按钮，删除这一行，操作示意图如下：

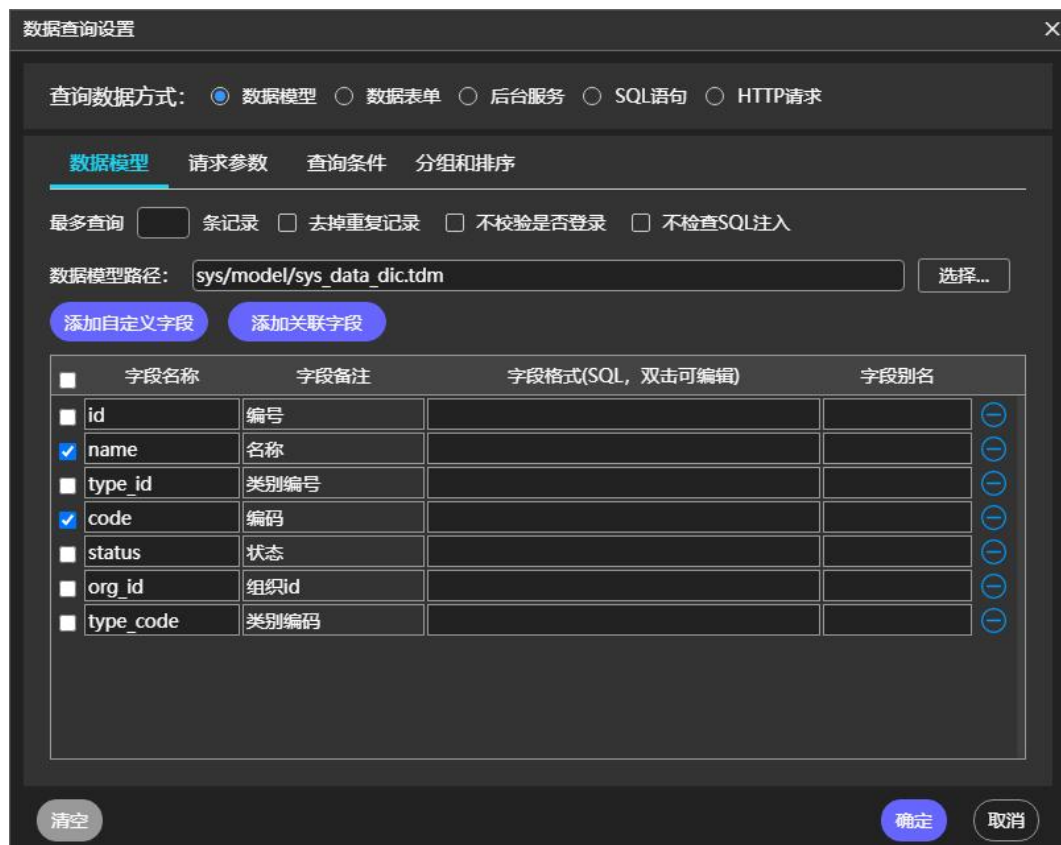


然后从组件库的只读组件中，拖拽“只读列表”组件到布局组件的左侧边栏的底部，并将其宽度设置为 100%，高度设置为 160px，如下图所示：



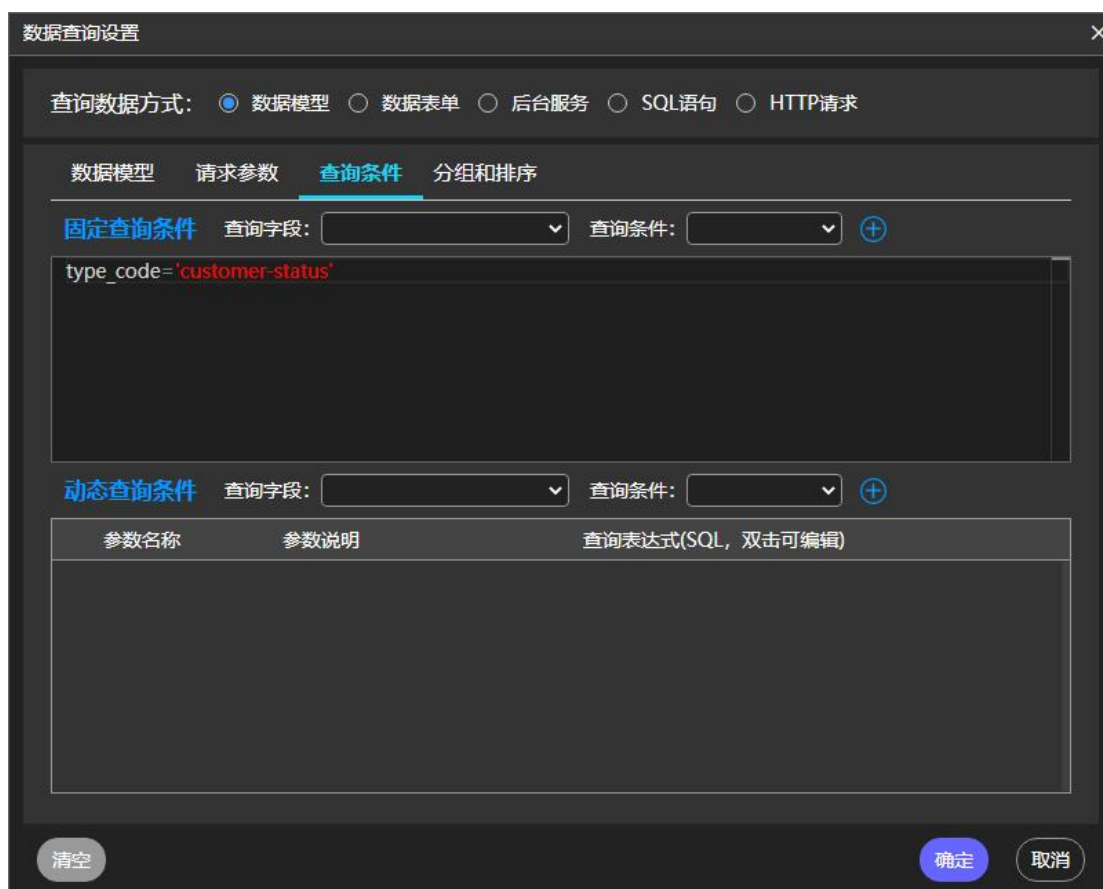


然后点击这个只读列表组件属性设置里的“数据查询设置”，数据模型选择 sys 项目里的 sys\_data\_dic.tdm（系统数据字典表），字段选择 name 和 code，如下图所示：





然后点击“查询条件”选项卡，在此将固定查询条件设置为：`type_code='customer-status'`，表示查询类别编码为“customer-status”的数据字典，如下图所示：



然后点确定，然后继续设置该列表组件的以下属性：

- **组件编号：** 设置为 status，要跟删除的那个客户状态查询输入项的编号一致；
- **选项值数据格式：** 设置为 code，表示列表值为客户状态数据字典的编码；
- **选项文本数据格式：** 设置为 name，表示列表文本为客户状态数据字典的名称；

然后保存页面，再点击预览按钮打开预览窗口，可以看到如下图所示界面，在页面左侧底部显示了一个列表，里面是所有可选的客户状态，点击某个状态，右侧列表里就会显示这个状态的所有客户：



### 3.8.6 修改列表页数据删除设置

在前面用向导创建的客户信息列表页面里，点击某一行数据最后面的删除按钮时，会将该行数据记录直接从数据库中删除，这对于客户这种重要信息来说，不太妥当，万一操作失误，要再找回就比较麻烦。建议改成逻辑删除，即在删除时，不真的删除该条数据记录，只是将其状态修改为“已删除”，并且记录删除人和删除时间，以便后续查证。具体操作方法如下：

在客户信息列表页面里，选中数据表格组件，然后点击属性里的“数据删除设置”，在打开的对话框中，在下面的“删除方式”中，选择“逻辑删除”，并在下面的字段列表设置要修改的三个字段及要修改的值：

- **状态**：不选择任何系统宏变量，直接添加，然后设置为-1，表示已删除。
- **删除人编号**：在系统宏变量里选择当前用户编号，对应的 JS 表达式为：req.session.userId；
- **删除时间**：在系统宏变量里选择当前日期时间，对应的 JS 表达式为：new Date()；

设置完后的界面如下图所示，然后点确定，保存列表页面，再删除就是逻辑删除了：

数据删除设置

删除数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型

请求参数

☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

选择...

删除条件 可选字段： 判断条件：

+

id={req.id}

删除方式：☐ 物理删除 ☒ 逻辑删除

字段列表： 系统宏变量：

+

| 字段名称        | 字段备注  | 字段值                |              |
|-------------|-------|--------------------|--------------|
| status      | 状态    | '-1'               | <div>⊖</div> |
| deletor_id  | 删除人编号 | req.session.userId | <div>⊖</div> |
| delete_time | 删除时间  | new Date()         | <div>⊖</div> |

清空

确定

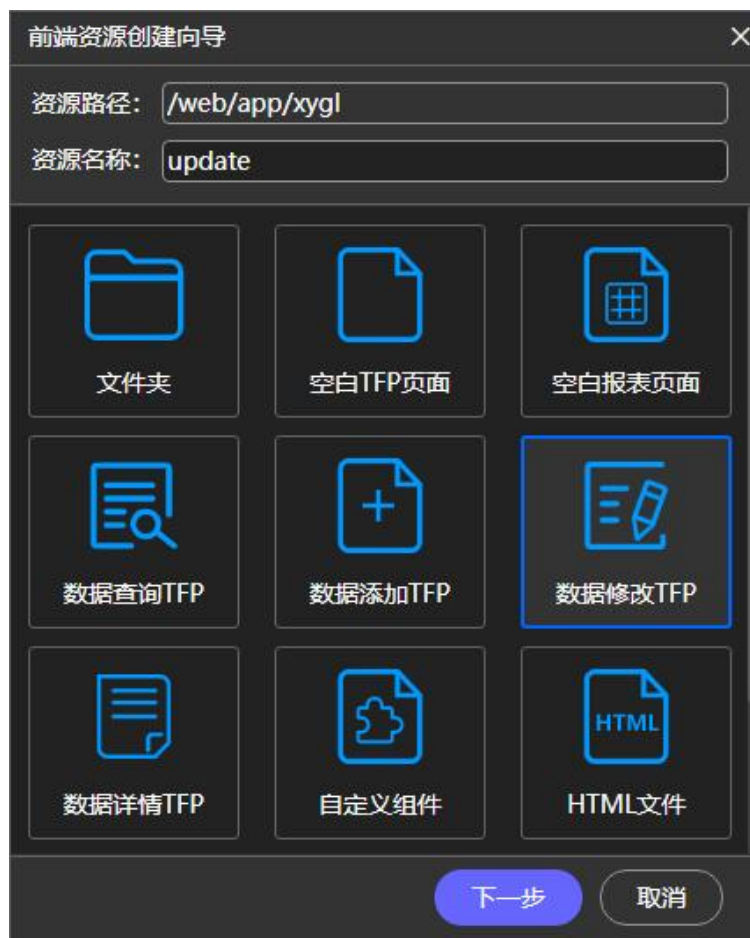
取消

## 3.9 创建客户信息修改页面

下面，我们再带领大家创建客户信息修改页面，具体的操作如下：

### 3.9.1 选择页面类型

点击示例项目前端文件里 customer 文件夹右侧的加号，打开“前端资源创建向导”对话框，选中“数据修改 TFP”，资源名称会自动设置为 update，这里不用改，使用默认的即可。



然后点击下一步，可以打开“创建数据修改 TFP”向导对话框。

### 3.9.2 设置页面参数

在“创建数据修改 TFP”向导对话框里，首先要选择数据模型，这里就选择前面创建的客户信息的数据模型，其他参数与前面介绍的“创建数据添加 TFP”向导里的基本一致，已经介绍过的参数就不再复述，这里不一样的地方就是需要设置一下数据查询服务，用来查询要修改的数据，可以是自动创建，也可以使用已有服务，如果要使用已经存在的服务，就点击右侧的“选择...”按钮进行选择，另外还可以设置查询服务返回的响应参数名，默认为：data。

然后在下面的“前端页面设置字段”列表里，删除不需要修改的字段，选中需要修改的字段，调整好显示顺序，并设置好各项参数，需要特殊设置的字段及特殊值如下表所示：

| 字段名称        | 输入项名称 | 输入项类型 | 数据格式              | 列数 | 数据字典            |
|-------------|-------|-------|-------------------|----|-----------------|
| status      |       |       |                   |    | customer-status |
| salesman_id | 销售员   | 用户选择  |                   |    |                 |
| scale       |       |       |                   |    | customer-scale  |
| grade       |       |       |                   |    | customer-grade  |
| source      |       |       |                   |    | customer-source |
| trade       |       |       |                   |    | customer-trade  |
| site        |       |       |                   | 2  |                 |
| province    | 所属省市  | 地区选择  | {province}/{city} |    |                 |
| city        |       | 隐藏域   |                   |    |                 |
| address     |       |       |                   | 2  |                 |
| comment     |       | 多行输入框 |                   | 3  |                 |

创建数据修改TFP

保存路径: /web/app/demo\_sm/customer/update.tfp

数据模型: demo\_sm/model/demo\_sm\_customer.tdm

选择...

页面标题: 修改客户信息

页面类型: 抽屉页

窗口大小: 宽

高

终端类型: 电脑端

背景颜色: 浅色

定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧

每行显示输入项数量: 3

仅支持PC

数据查询服务: ☒ 自动创建 ☐ 使用已有服务

响应参数名: data

选择...

数据修改服务: ☒ 自动创建 ☐ 使用已有服务

选择...

前端页面设置字段

后台服务设置字段

| <input checked="" type="checkbox"/> | 字段名称        | 输入项名称 | 输入项类型 | 数据格式          | 列数 | 必填                                  | 可选值 | 数据字典(双击选择)      |             |
|-------------------------------------|-------------|-------|-------|---------------|----|-------------------------------------|-----|-----------------|-------------|
| <input checked="" type="checkbox"/> | code        | 编码    | 单行输入框 | {code}        |    | <input checked="" type="checkbox"/> |     |                 | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |
| <input checked="" type="checkbox"/> | name        | 名称    | 单行输入框 | {name}        |    | <input checked="" type="checkbox"/> |     |                 | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |
| <input checked="" type="checkbox"/> | status      | 状态    | 单行输入框 | {status}      |    | <input checked="" type="checkbox"/> |     | customer-status | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |
| <input checked="" type="checkbox"/> | salesman_id | 销售员   | 用户选择  | {salesman_id} |    | <input checked="" type="checkbox"/> |     |                 | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |
| <input checked="" type="checkbox"/> | contacts    | 联系人   | 单行输入框 | {contacts}    |    | <input checked="" type="checkbox"/> |     |                 | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |
| <input checked="" type="checkbox"/> | telephone   | 联系电话  | 单行输入框 | {telephone}   |    | <input checked="" type="checkbox"/> |     |                 | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |
| <input checked="" type="checkbox"/> | scale       | 客户规模  | 单行输入框 | {scale}       |    | <input checked="" type="checkbox"/> |     | customer-scale  | ⤴ ⤵ ⤴ ⤵ ⤴ ⤵ |

确定

取消



创建数据修改TFP

保存路径: /web/app/demo\_sm/customer/update.tfp 数据模型: demo\_sm/model/demo\_sm\_customer.tdm 选择...

页面标题: 修改客户信息 页面类型: 抽屉页 窗口大小: 宽 高

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧 每行显示输入项数量: 3 仅支持PC

数据查询服务: ☒ 自动创建 ☐ 使用已有服务 响应参数名: data 选择...

数据修改服务: ☒ 自动创建 ☐ 使用已有服务 选择...

前端页面设置字段 后台服务设置字段

| 字段名称                                         | 输入项名称 | 输入项类型 | 数据绑定格式            | 列数 | 必填                                  | 可选值 | 数据字典(双击选择)      |
|----------------------------------------------|-------|-------|-------------------|----|-------------------------------------|-----|-----------------|
| <input checked="" type="checkbox"/> grade    | 客户等级  | 单行输入框 | {grade}           | 1  | <input checked="" type="checkbox"/> |     | customer-grade  |
| <input checked="" type="checkbox"/> source   | 客户来源  | 单行输入框 | {source}          | 1  | <input checked="" type="checkbox"/> |     | customer-source |
| <input checked="" type="checkbox"/> trade    | 所属行业  | 单行输入框 | {trade}           | 1  | <input checked="" type="checkbox"/> |     | customer-trade  |
| <input checked="" type="checkbox"/> site     | 客户网址  | 单行输入框 | {site}            | 2  | <input type="checkbox"/>            |     |                 |
| <input checked="" type="checkbox"/> province | 所在省市  | 地区选择  | {province}/{city} | 1  | <input checked="" type="checkbox"/> |     |                 |
| <input checked="" type="checkbox"/> city     | 所在城市  | 隐藏域   | {city}            | 1  | <input checked="" type="checkbox"/> |     |                 |
| <input checked="" type="checkbox"/> address  | 详细地址  | 单行输入框 | {address}         | 2  | <input type="checkbox"/>            |     |                 |
| <input checked="" type="checkbox"/> comment  | 备注    | 多行输入框 | {comment}         | 3  | <input type="checkbox"/>            |     |                 |

确定 取消

最终的设置界面如上面两图所示, 设置完毕后点击确定按钮, 即可创建客户信息修改页面。

其中, city 字段采用隐藏域是因为 province 采用的地区选择组件里已包含城市, 只需查询。

显示和定位 字体和文本 边距和边框 背景

透明度: % 内容溢出时: 光标样式:

组件库 组件大纲 K page1 属性 事件

容器组件

- 布局
- 面板
- 弹性面板
- 弹性栅格
- 表单
- 数据表格
- 可编辑表格
- 选项卡
- 数据卡片
- 树形组件
- 折叠面板
- 嵌入页面
- 宫格
- 宫格行
- 宫格列
- 单元

编码: \* 名称: \* 状态: \*

销售员: \* 联系人: \* 联系电话: \*

客户规模: \* 客户等级: \* 客户来源: \*

所属行业: \* 客户网址:

所在省市: \* 详细地址:

备注:

基本信息

- 组件编号: page1
- 组件类型: Page
- CSS样式类:
- 自定义CSS样式:

页面信息

- 终端类型: 电脑端
- 移动端类型:
- 组件默认定位方式: 浮动定位
- 前端UI框架: tfp
- 页面类型:

UI界面 TFP源码 JS脚本 CSS样式 HTML预览

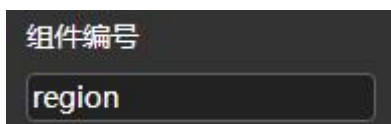


如果修改时，需要在后台设置某些字段的值，则可以点击向导对话框中的“后台服务设置字段”选项卡，然后在里面添加需要设置的字段及字段值，具体操作可以参考添加客户信息。

### 3.9.3 修改所属省市字段设置

前面介绍客户信息添加页面时讲过，在表单的数据提交设置里，需要对所在省份和所在城市这两个字段进行特殊处理，客户信息修改页面也是一样，具体的设置方法如下：

(1) 首先，在客户信息修改页面里选中所在省市对应的地区选择组件，在属性设置里，将该组件的编号改成：region，如下图所示：



(2) 然后，在页面里选中表单组件 form1，点击属性设置里面的“数据提交设置”，打开如下图所示对话框：

A screenshot of a '数据提交设置' (Data Submission Settings) dialog box. The '提交数据方式' (Data Submission Method) is set to '数据模型' (Data Model). The '提交类型' (Submission Type) is set to '修改' (Modify). The '数据模型路径' (Data Model Path) is 'demo\_sm/model/demo\_sm\_customer.tdm'. A table lists fields to be submitted, with 'province' and 'city' highlighted by a red box. The table has columns for '字段名称' (Field Name), '字段备注' (Field Remark), '字段值' (Field Value), and '主键' (Primary Key).

| 字段名称     | 字段备注 | 字段值                      | 主键 |
|----------|------|--------------------------|----|
| scale    | 客户规模 | req.scale                |    |
| grade    | 客户等级 | req.grade                |    |
| source   | 客户来源 | req.source               |    |
| trade    | 所属行业 | req.trade                |    |
| site     | 客户网址 | req.site                 |    |
| province | 所在省份 | req.region.split('/')[0] |    |
| city     | 所在城市 | req.region.split('/')[1] |    |
| address  | 详细地址 | req.address              |    |
| comment  | 备注   | req.comment              |    |

在该对话框中，将所在省份和所在城市分别设置为以下值：

- **province:** req.region.split('/')[0] //表示从前端页面所在省市输入项的值里取/前面的编码
- **city:** req.region.split('/')[1] //表示从前端页面所在省市输入项的值里取/后面的编码

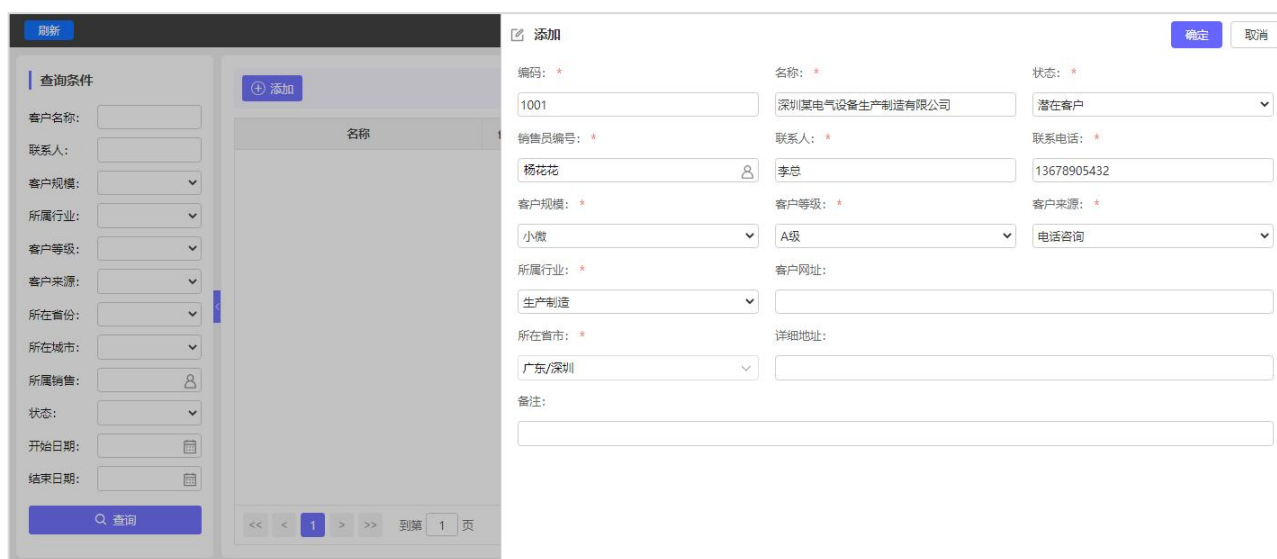
设置好之后，点确定按钮关闭对话框，然后点击工具栏上的保存按钮保存当前页面的修改，这时候，就会自动重新生成修改客户信息的后台服务，然后就可以预览测试了。

### 3.9.4 测试客户信息修改页面

现在我们用预览工具测试一下客户信息修改页面是否能正确运行。

因为客户信息修改页面需要知道具体要修改哪个客户的信息，所以必须在列表页面里点击某个客户信息所在行的修改按钮打开修改页面，不能直接预览修改页面进行测试。

在 TaskBuilder 里打开前面创建的客户信息列表页面，然后点击工具栏上的预览按钮，会在预览工具里打开客户信息列表页面，点击表格里某个客户所在行的修改按钮，即可以看到从界面右侧会滑出一个抽屉页面，里面显示的内容就是我们创建的客户信息修改页面，整体界面如下图所示：



在该抽屉页中可以修改客户有关的信息，然后点击右上角的确定按钮，即可实现客户信息的修改，并且会自动关闭抽屉页，然后刷新列表页面里的数据表格组件，重新加载数据。

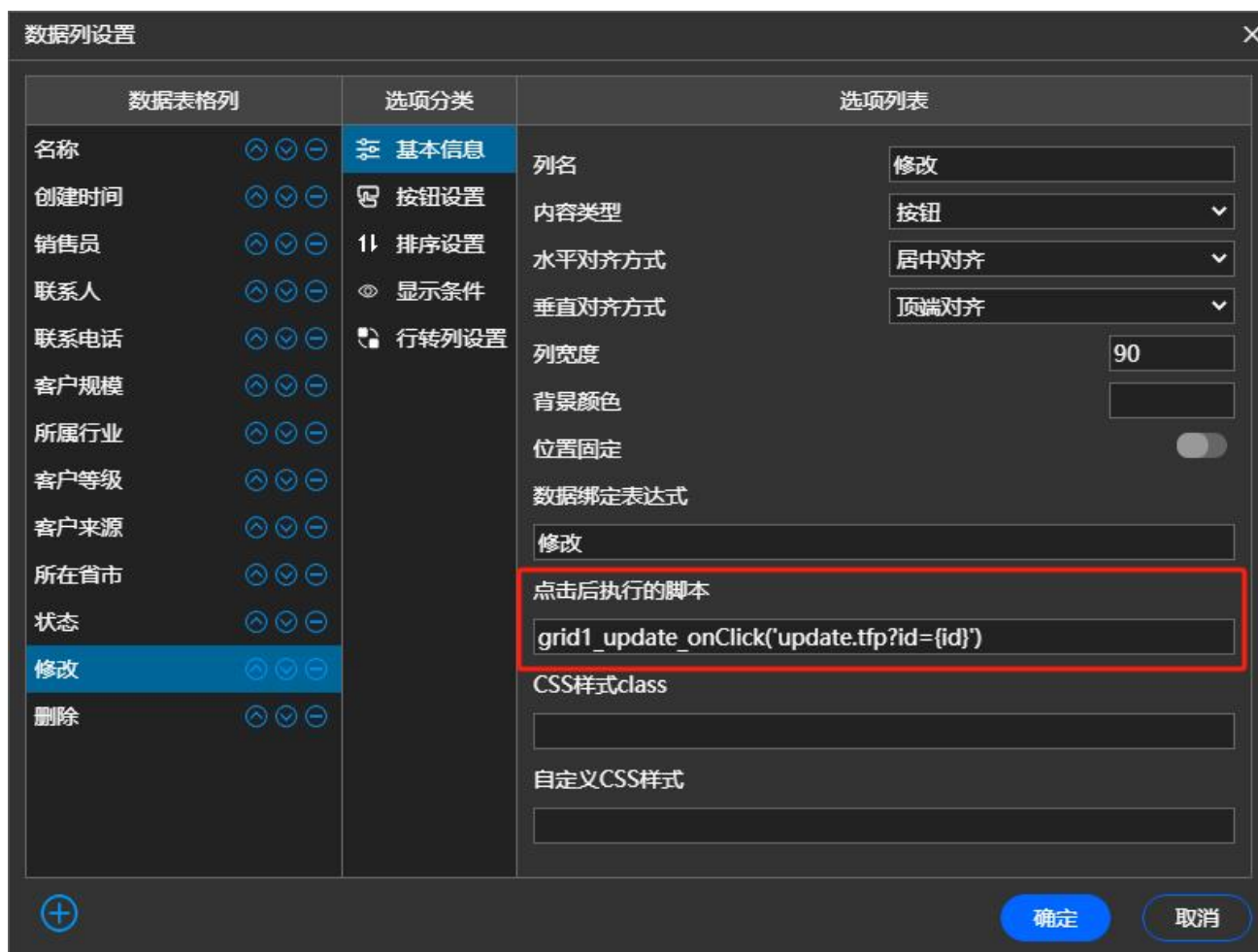
## 3.10 客户信息修改页实现原理解析

整个客户信息修改页面的功能分为以下几步来实现：

### 3.10.1 打开客户信息修改页面

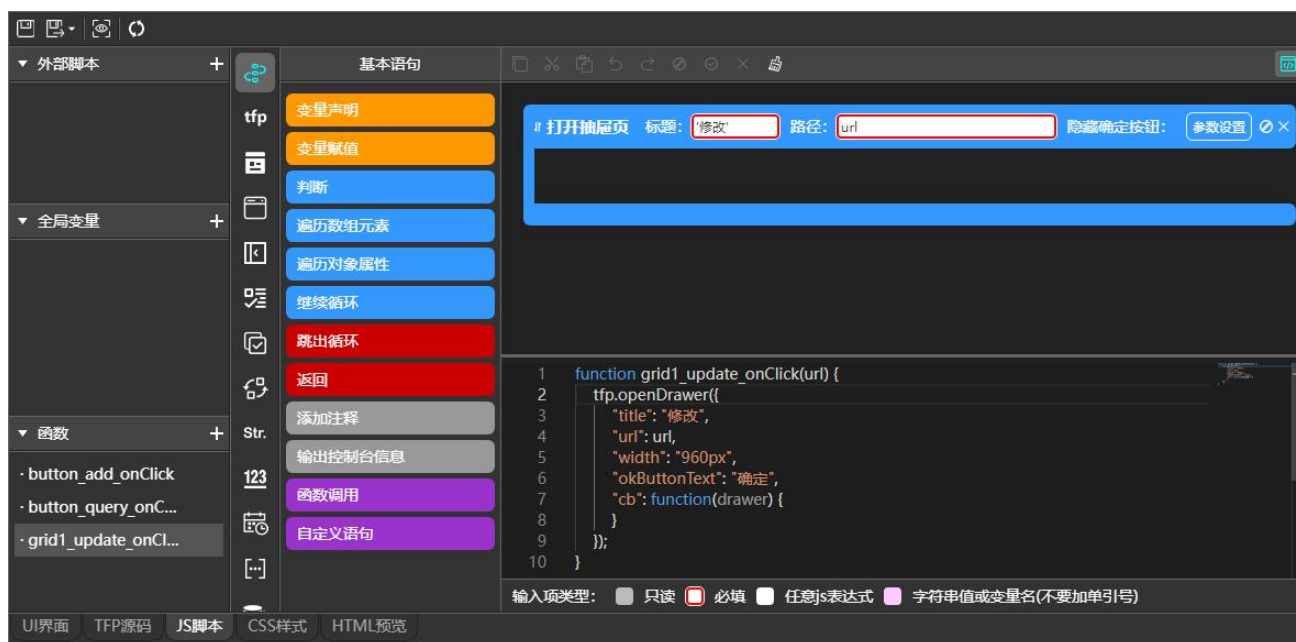
在客户信息列表页面的数据表格里，点击修改按钮时，需要打开修改页面，并传递当前行的客户编号（id），设置方法如下：

在客户信息列表页面中，选中数据表格组件，然后点击属性里的“数据列设置”，在打开的对话框中，选中右侧列表里的“修改”这一列，然后可以看到在“点击后执行的脚本”里设置了如下图所示的脚本，这段脚本的意思是，点击修改按钮后，执行名为：grid1\_update\_onClick 的脚本函数，并传入修改页面的 URL 路径，里面传递的 id 参数就是当前行的客户编号。



关闭该对话框，点击列表页面底部的“JS 脚本”，然后在函数列表里点击函数：

grid1 update onClick, 可以看到该函数的脚本设置如下图所示:



从图中可知，执行该函数时，会打开一个抽屉页，该页面的地址就是传给该函数的 URL 参数，也就是上面设置的修改页面的地址，加上要传给修改页面的客户编号参数（id）。

### 3.10.2 查询要修改的数据

打开修改页面后，就需要先查询需要修改的数据，实现方法如下：

选中客户信息修改页面里的表单组件 form1，点击属性里的“数据查询设置”，会打开如下所示的对话框，在此可以看到，向导已经为我们设置好表单组件的数据查询相关参数，包括要查询的数据模型以及要查询的字段等：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型

请求参数

查询条件

分组和排序

☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

选择...

添加自定义字段

添加关联字段

| <input type="checkbox"/>            | 字段名称        | 字段备注  | 字段格式(SQL, 双击可编辑) | 字段别名 |   |
|-------------------------------------|-------------|-------|------------------|------|---|
| <input checked="" type="checkbox"/> | id          | 编号    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | code        | 编码    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | name        | 名称    |                  |      | ⊖ |
| <input type="checkbox"/>            | creator_id  | 创建人编号 |                  |      | ⊖ |
| <input type="checkbox"/>            | create_time | 创建时间  |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | status      | 状态    |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | salesman_id | 销售员编号 |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | contacts    | 联系人   |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | telephone   | 联系电话  |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | scale       | 客户规模  |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | grade       | 客户等级  |                  |      | ⊖ |

清空

确定

取消

另外，点击上面的“请求参数”选项卡，可以看到里面设置了一个名为 id 的参数：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型

请求参数

查询条件

分组和排序

+

↓ 在此设置需要传递给后台的查询参数

| 参数名 | 参数类型  | 参数值 |                    |
|-----|-------|-----|--------------------|
| id  | URL参数 |     | <div>⊕ ⊖ ⊡ ⊢</div> |

清空

确定

取消

设置该参数的目的就是在打开修改页面时要从 URL 路径里获取要修改的客户编号 (id) , 并传递给查询客户信息的后台服务。

再点击“查询条件”选项卡, 可以看到下图所示界面:

数据查询设置

查询数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 **查询条件** 分组和排序

**固定查询条件** 查询字段:  查询条件:  +

id={req.id}

**动态查询条件** 查询字段:  查询条件:  +

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

清空 确定 取消

在此可以看到, 在后台执行真正的数据查询操作时, 会用客户信息表里的 id 字段关联前端页面发送的请求参数里的 id 参数的值进行查询, 这样就只会查询到该编号对应的客户信息。

通过上面的设置, 打开修改页面时, 就会自动查询对应的客户信息。

### 3.10.3 将数据绑定到输入项内

在修改页面内, 选中某个输入项, 例如客户名称输入项 (组件编号为: name) , 在属性设置里, 就可以看到该输入项的“数据绑定格式”设置为了: {name}, 这样设置就可以实现在打开修改页面时, 将查询到的客户信息里的 name 字段的值绑定到该输入项内并显示出来。



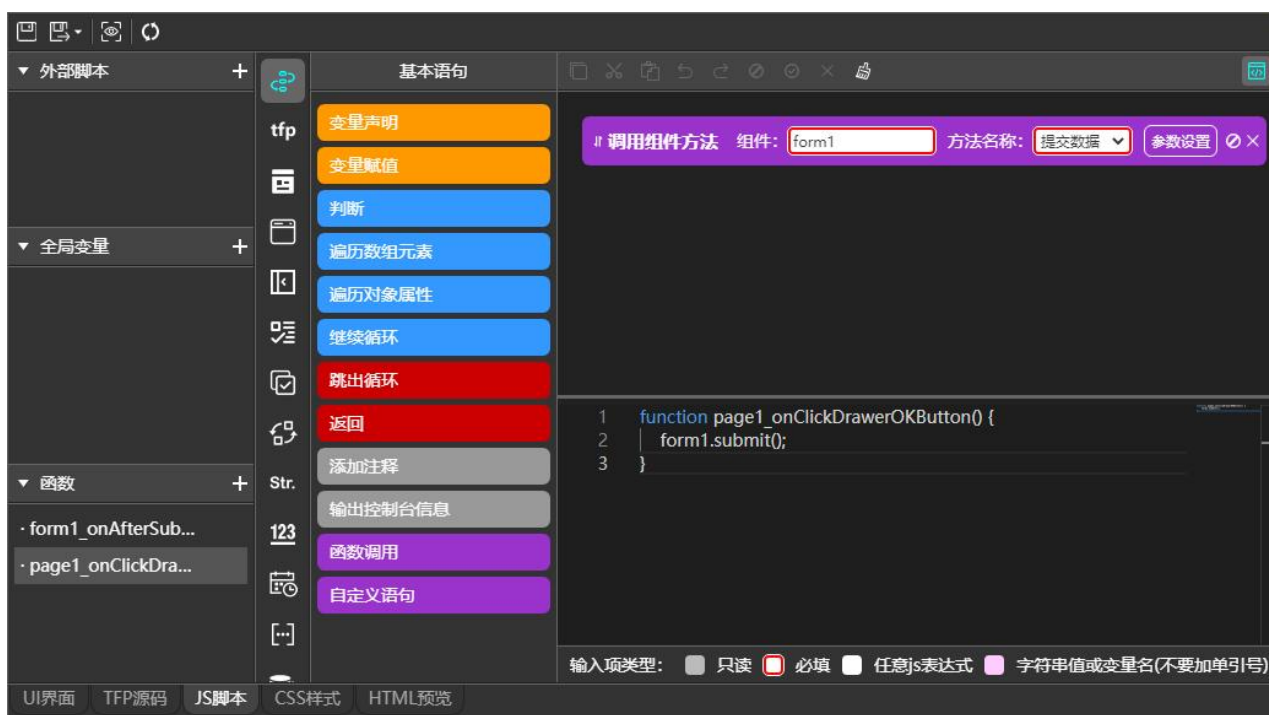
其他输入项也都是一样的道理，其中有几种特殊的输入项，他们绑定的值是编号或编码，不是最终显示的文字，包括：

- **用户选择**：数据绑定格式为 {用户编号}，例如销售员编号：{salesman\_id}，绑定时会自动根据编号查询用户的姓名并显示；
- **数据字典**：数据绑定格式为 {字典编码}，例如客户状态：{status}，绑定时会自动根据字典编码查询字典名称并显示；
- **地区选择组件**：数据绑定格式为 {省份代码}/{城市代码}，例如：{province}/{city}，绑定时会自动查询省份名称和城市名称并显示。

### 3.10.4 提交修改完的数据

用户在修改完数据后，会点击抽屉右上角的“确定”按钮提交数据，完成客户信息的修改，该操作的实现原理如下：

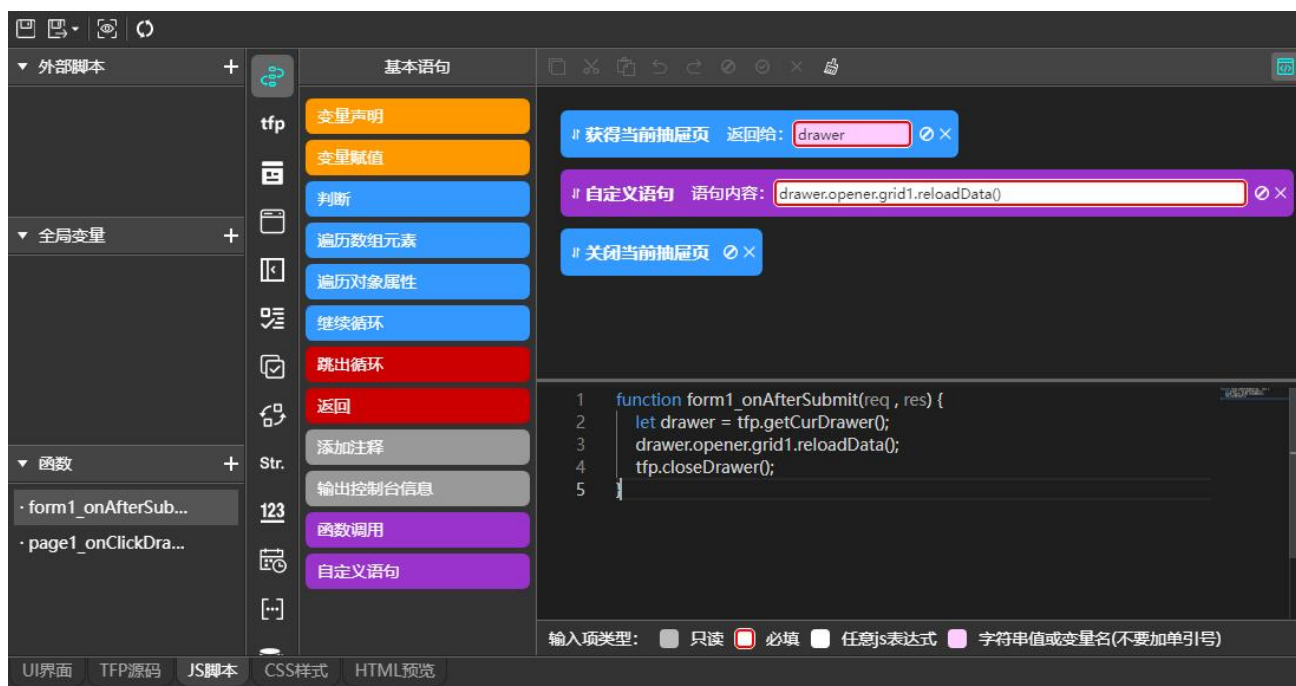
在客户信息修改页面里，选中页面组件（page1），然后双击事件设置面板里的“点击抽屉页确定按钮时”事件输入框，打开该事件处理函数的设置界面，如下图所示：



从上图可知，点击抽屉页的确定按钮后，会调用表单组件 form1 的 submit()方法提交数据。

### 3.10.5 刷新客户列表，关闭抽屉页

再回到修改页面的“UI 界面”设计器里，选中表单组件 form1，然后双击事件设置面板里的“提交后”事件输入框，打开该事件处理函数的设置界面，如下图所示：



从上图可知，当完成表单的数据提交后，会让列表页面里的数据表格组件重新加载数据，然后关闭当前抽屉页。

## 3.11 发布客户管理应用

在开发完一个独立的应用功能后，可以将其发布到后台管理系统中，操作方法如下：

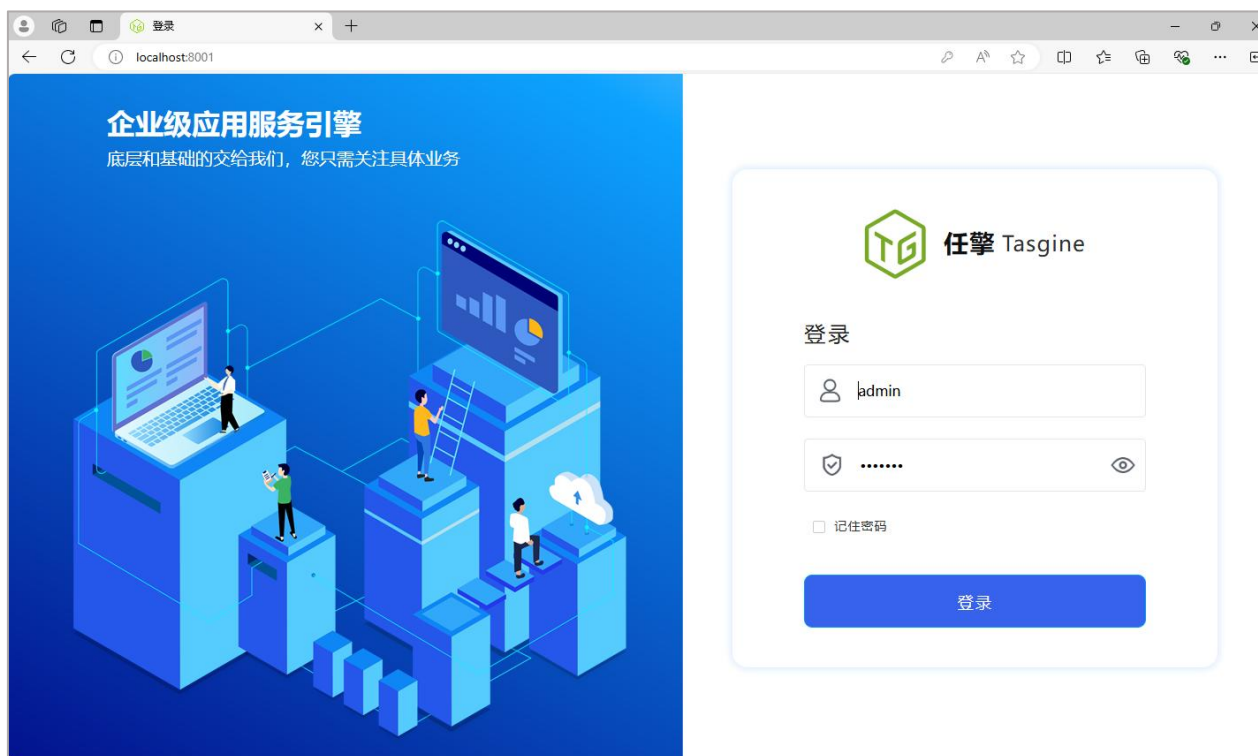
### 3.11.1 登录后台管理系统

有两种方法可以进入后台管理系统：

#### (1) 在浏览器里输入后台管理系统地址从登录页面进入

打开浏览器（推荐 Chrome），在地址栏输入地址：http://127.0.0.1:8001，然后按回车，

即可打开任擎后台管理系统的登录页面，如下图所示：



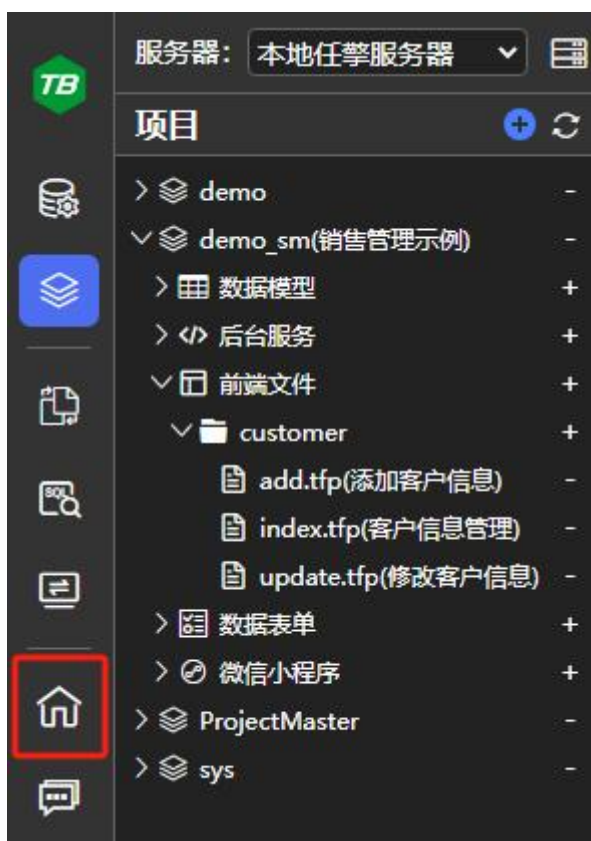
如果任擎安装在别的电脑上或者端口不是 8001,则需要修改上面路径里的 IP 地址和端口。

在此页面内输入用户名：admin（后台管理员账号）和密码：taskmsg（默认密码），然后点登录，即可进入后台管理系统，如下图所示：



## (2) 点击 TaskBuilder 系统侧边栏上的“后台管理系统”图标进入

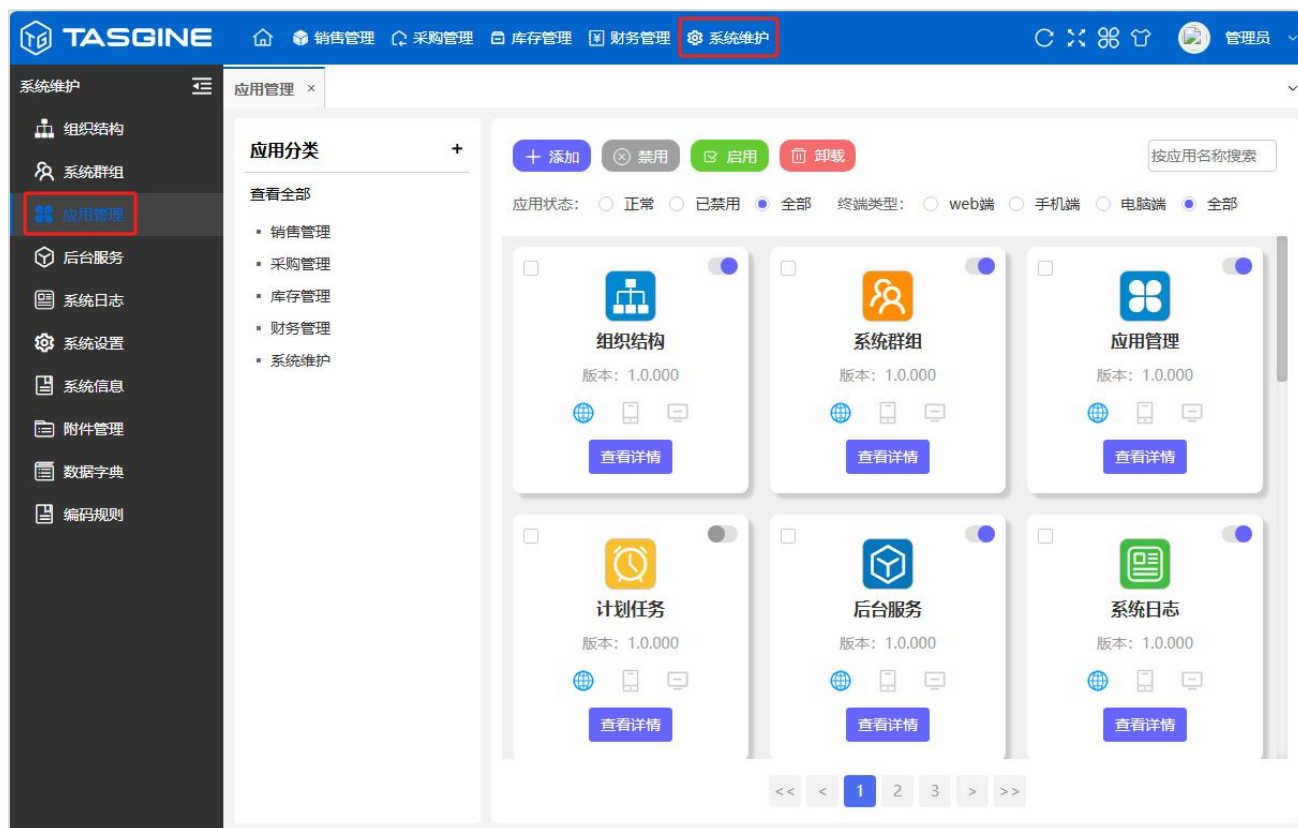
在 TaskBuilder 中，点击如下图所示的左侧系统侧边栏里的“后台管理系统”图标，也可以直接登录任擎后台管理系统，不需要再输入账号和密码，因为在 TaskBuilder 的服务器设置里，已经设置了相关信息。



### 3.11.2 添加应用分类

我们带领大家开发的示例项目包含客户管理、产品管理、销售订单、销售统计等多个应用，为了方便管理，可以将这些应用都放到一个大模块内，这时候，就需要用到应用分类的功能。

在后台管理系统内，点击顶部一级功能导航栏里的“系统维护”，再点击左侧菜单栏里的“应用管理”，会打开应用列表页面，如下图所示：



在该页面中，左侧的应用分类相当于存放应用的文件夹，可以支持多级分类。

点击左侧应用分类标题栏右边的加号按钮，可以打开添加应用分类的对话框，如下图所示：



在此输入类别代码：demo\_sm 和类别名称：示例项目，父类别不用选，因为我们要建的



分类是一级分类，然后点击类别图标设置的“选择”按钮，打开图标选择对话框，如下图所示：



在左边先选择一个图标，然后在右边设置好背景颜色，然后点保存按钮，然后再点确定按钮，即可完成应用分类的创建，如下图所示：



### 3.11.3 添加应用信息

在应用管理页面内，点击应用列表上部的“添加”按钮，可以打开“添加应用”的页面，如下图所示：



### 添加应用

应用图标: \*

选择



应用代码: \*

应用名称: \*

所属类别: \*

当前版本: \*

1.0.0.1

支持终端: \*

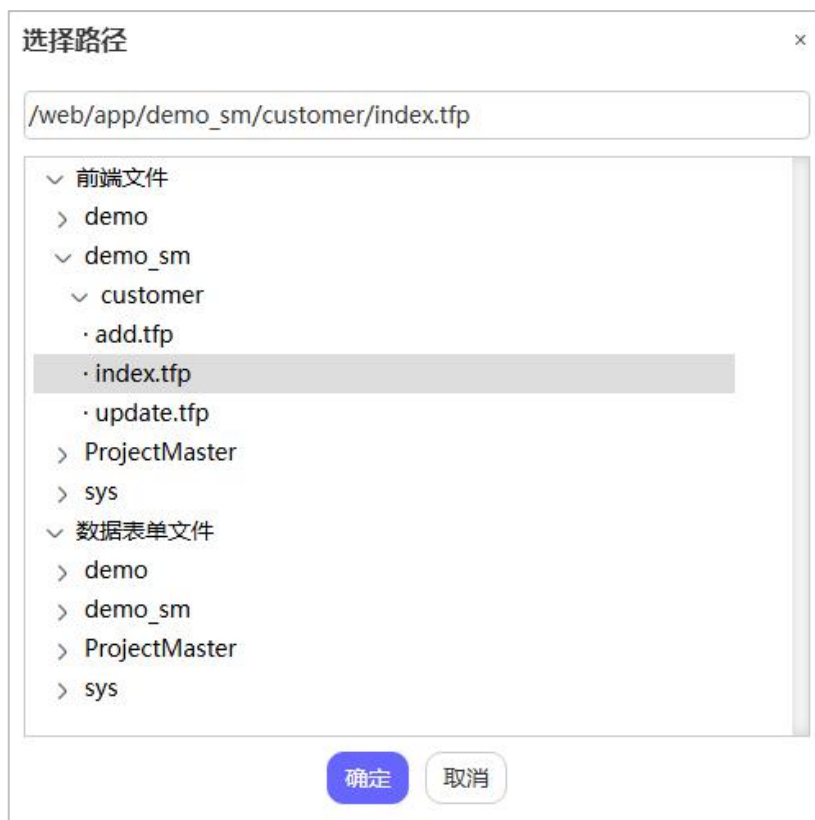
☐ web ☐ 任讯手机客户端 ☐ 任讯电脑客户端

描述:

提交

在此需要设置以下信息：

- **应用图标：**设置方法与上面设置应用分类的方法一致；
- **应用代码：**设置为 demo\_sm\_customer；
- **应用名称：**设置为客户管理；
- **所属类别：**选择前面创建的“示例项目”这个类别；
- **支持终端：**选择 Web；
- **web 端首页 URL 路径：**点击右边的“选择”按钮，在弹出的对话框中选中客户管理模块的首页：前端文件 > demo\_sm > customer > index.tfp 文件，然后点确定按钮；



都设置好之后的界面如下图所示：

添加应用

应用图标：\* 选择



应用代码：\* demo\_sm\_customer

应用名称：\* 客户管理

所属类别：\* 示例项目 ▼

当前版本：\* 1.0.0.1

支持终端：\* ☒ web ☐ 任讯手机客户端 ☐ 任讯电脑客户端

web端首页URL路径：\* /app/demo\_sm/customer/index.tfp 选择

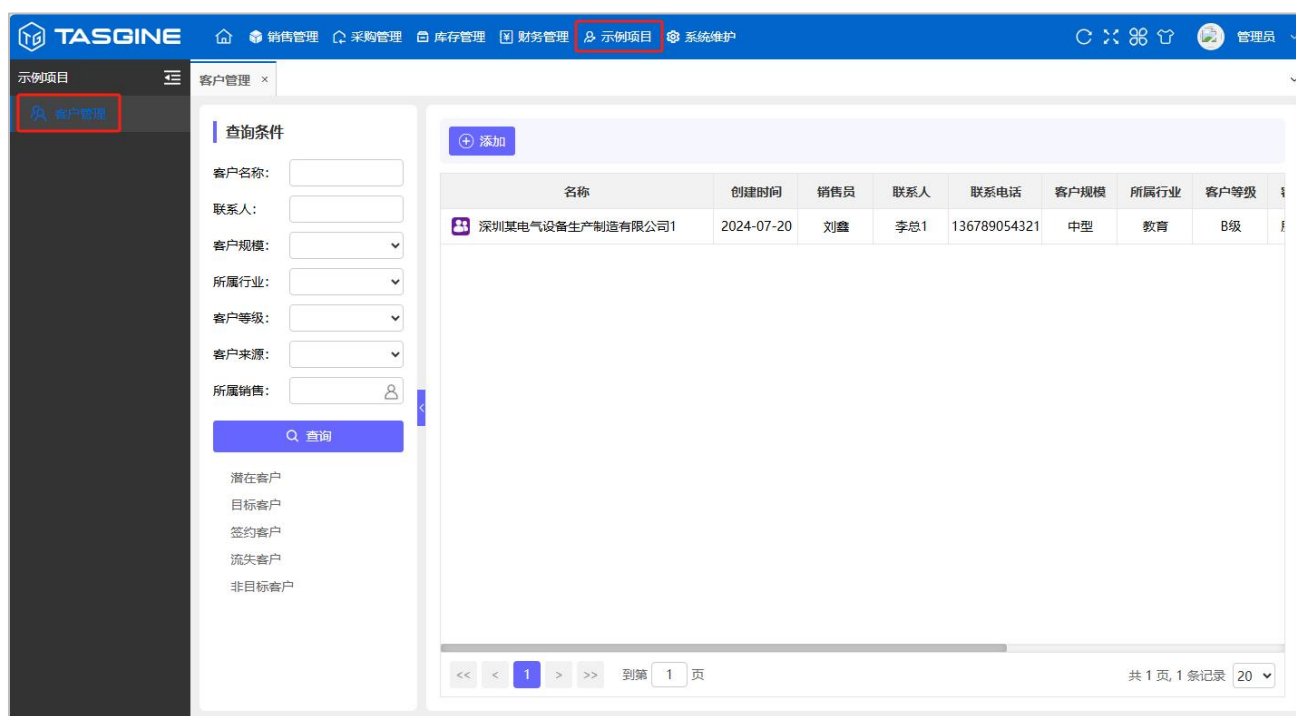
web端详细URL路径： 选择

描述：

提交

点击底部的“提交”按钮，即可完成客户管理应用的添加。

然后点击浏览器里的刷新按钮，刷新整个后台管理系统，就可以在顶部功能导航栏里看到“示例项目”，点击该模块，就可以在左侧菜单里看到“客户管理”模块，再点击该模块，就可以看到客户信息列表页面，如下图所示：



### 3.11.4 设置应用访问权限

新添加的应用，默认只有 admin 账号（系统管理员）登录后可以看到，如果想让其他用户或群组也能访问，需要设置该应用的访问权限，操作方法如下：

在任擎后台管理系统内，打开系统维护里的应用管理模块，点击左侧应用分类里的“示例项目”，再点击右边应用列表里“客户管理”应用的“查看详情”按钮，可以打开该应用的详细信息页面，如下图所示：

应用信息

▪ 基本信息

▪ 访问权限

▪ 操作管理

基本信息

应用图标: \* 

选择



应用代码: demo\_sm\_customer

应用名称: \*

所属类别: \* 

示例项目

序号: \*

当前版本: \*

支持终端: \* ☒ web端 ☐ 手机客户端 ☐ 电脑客户端

web端首页URL路径: \* 

选择

web端详细页URL路径: 

选择

描述:

保存

点击该页面左侧的访问权限，可以看到下图所示界面：

应用信息

▪ 基本信息

▪ 访问权限

▪ 操作管理

访问权限

添加

类型:

拥有者:

终端类型:

查询

| 类型   | 拥有者   | 终端类型 | 操作            |
|------|-------|------|---------------|
| 系统群组 | 系统管理员 | web  | <div>删除</div> |

<<

<

1

>

>>

到第  页

共 1 页, 1 条记录

20

在此可以看到，已为系统管理员添加了该应用 web 端的访问权限，如果还要给其他用户添加访问权限，则可以点击上部的“添加”按钮，打开如下图所示的“添加权限”对话框，在该对话框中，权限的拥有者可以选择系统群组、部门和用户这三种类型，默认为系统群组，下面的列表里会列出系统里维护好的所有群组信息，可以选择一个或多个群组，点确定按钮添加权限之后，就表示这些群组里的用户都可以在系统菜单里看到这个应用，如果选择部门，则下面的列表里会显示系统里的所有部门，选择一个或多个部门，点确定按钮添加权限后，就表示北京任讯信息技术有限公司

这些部门里的人都可以在系统菜单里看到这个应用，如果选择用户，则下面的列表会按组织结构列出系统里的所有用户，选择一个或多个用户，点确定按钮添加权限后，这些用户就可以在系统菜单里看到这个应用。

另外，如果添加应用时，为应用配置了多个终端的首页地址，则在添加权限时，可以精细化设置该权限具体是指哪个或哪些终端。



如果要取消权限，则可以在上面的权限列表里，点该权限右侧的删除按钮删除该权限即可。

### 3.11.5 添加应用操作信息

上面设置的应用权限是指用户登录任擎后台管理系统后，是否可以在系统菜单里看到该应用，如果用户知道该页面的地址，直接在浏览器里输入该页面的地址，就可以绕过上面的应用权限控制直接访问该页面，因此，这种权限机制是不够安全的，为了加强系统的安全性，做到更精细化的权限控制，任擎后台管理系统还提供了应用操作权限管控的功能。

这里所说的应用操作就是指应用各个页面里用到的后台服务，应用操作权限就是指控制用户是否可以访问指定后台服务的权限，将某个后台服务添加到某个应用的操作列表里后，不管用户采用什么方式请求该后台服务，任擎都会进行权限判断，如果无权访问会报错。

在查看应用详情的页面里，点击左侧的“操作管理”，可以看到如下图所示界面：



点击操作列表上面的“添加”按钮，可以打开添加操作信息的对话框，如下图所示：



在此可以输入操作名称，然后点击服务路径右边的“选择”按钮，在弹出的服务选择对话框里选择该操作对应的后台服务，然后可以选择支持的终端，包括电脑端和手机端，默认是支持全部终端，表示从电脑和手机端都可以访问该服务，都设置好后，点确定按钮即可完成操作信息的添加。

客户管理应用涉及到的操作和对应的后台服务如下表所示：

| 操作名称     | 服务路径                                             |
|----------|--------------------------------------------------|
| 查询客户信息列表 | /demo_sm/service/customer/index_grid1_query.tbs  |
| 查询指定客户详情 | /demo_sm/service/customer/update_form1_query.tbs |



|        |                                                   |
|--------|---------------------------------------------------|
| 添加客户信息 | /demo_sm/service/customer/add_form1_add.tbs       |
| 修改客户信息 | /demo_sm/service/customer/update_form1_update.tbs |
| 删除客户信息 | /demo_sm/service/customer/index_grid1_delete.tbs  |

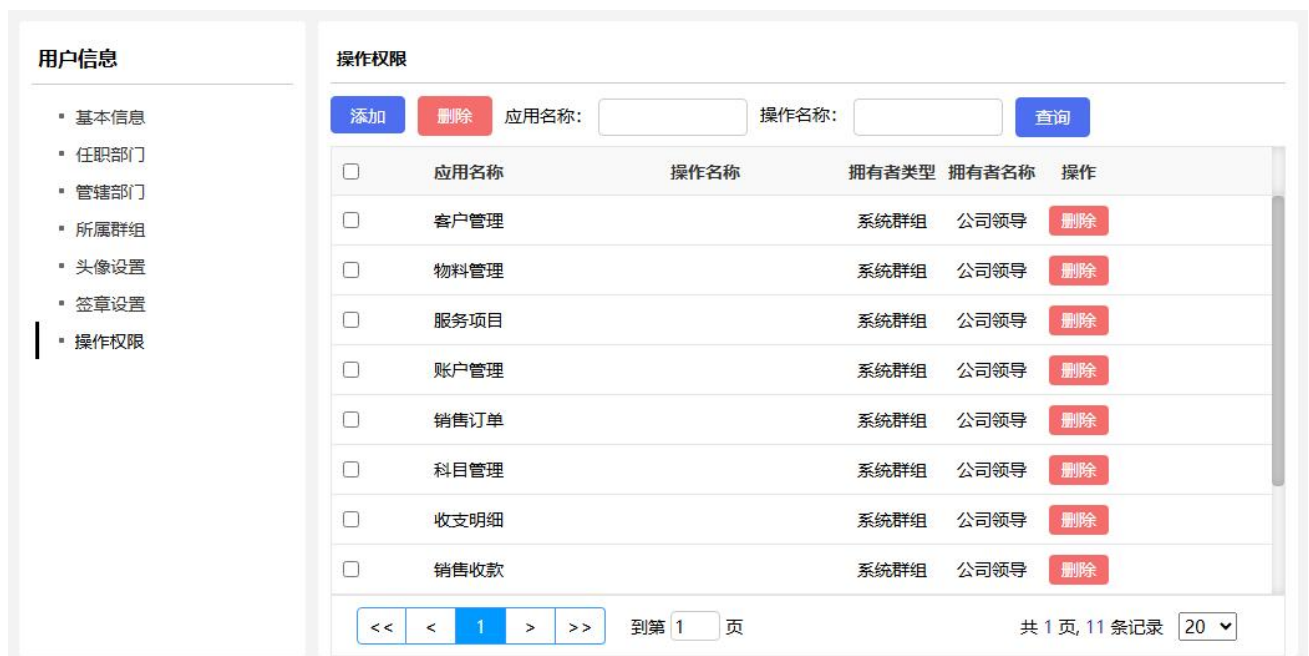
如果需要对这些操作进行严格的权限管控，则可以将他们都添加到“客户管理”应用的操作管理中，如果不添加，则默认所有登录用户都可以访问这些后台服务。

### 3.11.6 设置应用操作权限

在给应用添加完操作信息后，需要给相关用户或系统群组设置操作权限，这些用户和群组才能调用这些后台服务，否则会收到无权访问的错误提示，设置操作权限的方法如下：

#### 3.11.6.1 设置用户登录操作权限

在 系统维护 -> 组织结构 模块中，点击某个用户的姓名，在打开的用户详情页面里，点击左侧的“操作权限”，可以看到该用户可以访问的操作列表，如下图所示：



点击操作列表上面的“添加”按钮，打开添加操作权限的对话框，如下图所示：



在该对话框中选中的一个或多个操作，然后点确定按钮，即可完成添加，添加后，该用户就可以访问这些操作对应的后台服务。

### 3.11.6.2 设置系统群组的操作权限

如果有多个用户的权限是相同的，则可以创建一个系统群组，然后把这些用户添加到这个群组里，然后再设置这个群组的操作权限，设置完后，这个群组里的所有用户都会自动拥有执行这些操作的权限，设置方法如下：

打开 系统维护 -> 系统群组 模块，点击某个群组名称，打开该群组的详情页面，然后点左侧的“操作权限”，就可以看到该群组已经设置好的操作权限列表，如下图所示：



点击“添加”按钮，即可为该群组添加操作权限，具体操作参考上面为用户操作权限设置。

至此，我们就完成了客户管理模块的开发。

## 4、产品管理模块开发

接下来,我们讲解产品管理模块的开发,该模块用来实现产品分类信息和产品信息的管理,在介绍该模块的开发时,前面已经介绍过的功能就不再复述,重点讲解一些新的功能点。

### 4.1 本节知识点

- (1) 数据表格显示图片;
- (2) 对话框的使用;
- (3) 树形组件的使用;
- (4) 树形下拉列表的使用;
- (5) 图片上传组件的使用;
- (6) 用图片查看器查看大图;
- (7) 批量修改数据的实现
- (8) 在 SQL 中插入 JS 变量的两种方式
- (9) 导入导出组件的使用;

### 4.2 创建数据模型

产品管理模块需要创建产品分类和产品信息两个数据模型,具体信息如下:

#### 4.2.1 创建产品分类数据模型

产品分类数据模型包括以下字段:

| 字段名  | 数据类型 | 长度 | 字段备注 | 非空 | 唯一 | 说明      |
|------|------|----|------|----|----|---------|
| id   | 整数   | 11 | 分类编号 | 是  | 是  | 主键, 自增长 |
| code | 字符串  | 10 | 分类编码 | 是  |    |         |

|           |     |    |        |   |  |                       |
|-----------|-----|----|--------|---|--|-----------------------|
| name      | 字符串 | 20 | 分类名称   | 是 |  |                       |
| parent_id | 整数  | 11 | 上级分类编号 | 是 |  | 默认为 0，表示没有上级          |
| sort_num  | 整数  | 11 | 分类序号   |   |  |                       |
| status    | 微整数 | 4  | 分类状态   | 是 |  | 0 表示正常，-1 表示已禁用，默认为 0 |

请使用 TaskBuilder 的新建数据模型向导在 demo\_sm 示例项目的数据模型里，创建产品分类信息的数据模型：demo\_sm\_product\_type.tdm，具体操作就不再详细讲解，请参考客户管理模块里的相关介绍。

产品分类数据模型创建完的界面如下图所示：

| <input type="checkbox"/> | 字段名       | 字段类型 | 长度 | 精度 | 主键                                  | 非空                                  | 自增                                  | 唯一                       | 默认值 | 备注     |
|--------------------------|-----------|------|----|----|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-----|--------|
| <input type="checkbox"/> | id        | 整数   | 11 |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |     | 分类编号   |
| <input type="checkbox"/> | code      | 字符串  | 10 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 分类编码   |
| <input type="checkbox"/> | name      | 字符串  | 20 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 分类名称   |
| <input type="checkbox"/> | parent_id | 整数   | 11 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 上级分类编号 |
| <input type="checkbox"/> | sort_num  | 整数   | 11 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 分类序号   |
| <input type="checkbox"/> | status    | 微整数  | 4  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 分类状态   |

## 4.2.2 创建产品信息数据模型

请按下表要求创建产品信息数据模型，模型名称为：demo\_sm\_product.tdm：

| 字段名       | 数据类型 | 长度  | 字段备注   | 非空 | 唯一 | 说明               |
|-----------|------|-----|--------|----|----|------------------|
| id        | 整数   | 11  | 产品编号   | 是  | 是  | 主键，自增长           |
| code      | 字符串  | 10  | 产品编码   | 是  |    |                  |
| name      | 字符串  | 50  | 产品名称   | 是  |    |                  |
| type_id   | 整数   | 11  | 产品分类编号 | 是  |    | 关联产品分类查询名称       |
| sort_num  | 整数   | 11  | 产品序号   |    |    |                  |
| model     | 字符串  | 200 | 规格型号   |    |    |                  |
| unit      | 字符串  | 2   | 计量单位   | 是  |    | 关联计量单位数据字典       |
| price     | 金额   | 16  | 单价     |    |    | 精度为 2，值默认为 0.00  |
| thumbnail | 字符串  | 32  | 缩略图代码  |    |    | 关联附件表查询详情        |
| comment   | 字符串  | 200 | 备注     |    |    |                  |
| status    | 微整数  | 4   | 产品状态   | 是  |    | 0 表示正常，-1 表示已禁用， |

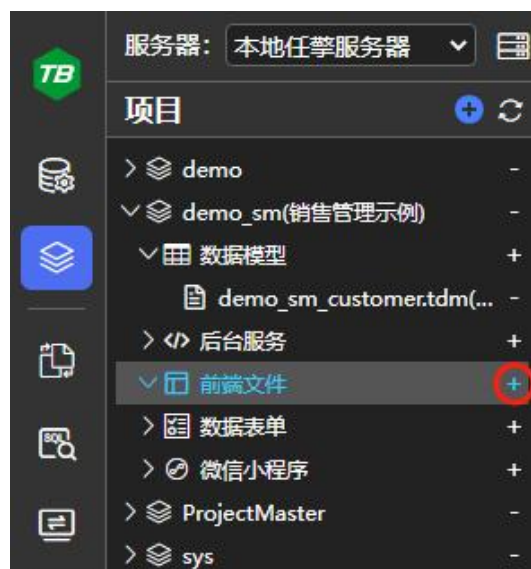
|             |      |    |      |  |  |       |
|-------------|------|----|------|--|--|-------|
|             |      |    |      |  |  | 默认为 0 |
| deletor_id  | 整数   | 11 | 删除人  |  |  |       |
| delete_time | 日期时间 |    | 删除时间 |  |  |       |

产品信息数据模型创建完的界面如下图所示：

| <input type="checkbox"/> | 字段名         | 字段类型 | 长度  | 精度 | 主键                                  | 非空                                  | 自增                                  | 唯一                       | 默认值 | 备注     |
|--------------------------|-------------|------|-----|----|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-----|--------|
| <input type="checkbox"/> | id          | 整数   | 11  |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |     | 产品编号   |
| <input type="checkbox"/> | code        | 字符串  | 10  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 产品编码   |
| <input type="checkbox"/> | name        | 字符串  | 50  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 产品名称   |
| <input type="checkbox"/> | type_id     | 整数   | 11  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 产品分类编号 |
| <input type="checkbox"/> | sort_num    | 整数   | 11  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 产品序号   |
| <input type="checkbox"/> | model       | 字符串  | 200 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 规格型号   |
| <input type="checkbox"/> | unit        | 字符串  | 2   |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 计量单位   |
| <input type="checkbox"/> | price       | 金额   | 16  | 2  | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 单价     |
| <input type="checkbox"/> | thumbnail   | 字符串  | 32  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 缩略图代码  |
| <input type="checkbox"/> | comment     | 字符串  | 200 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 备注     |
| <input type="checkbox"/> | status      | 微整数  | 4   |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 状态     |
| <input type="checkbox"/> | deletor_id  | 整数   | 11  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 删除人编号  |
| <input type="checkbox"/> | delete_time | 日期时间 |     |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 删除时间   |

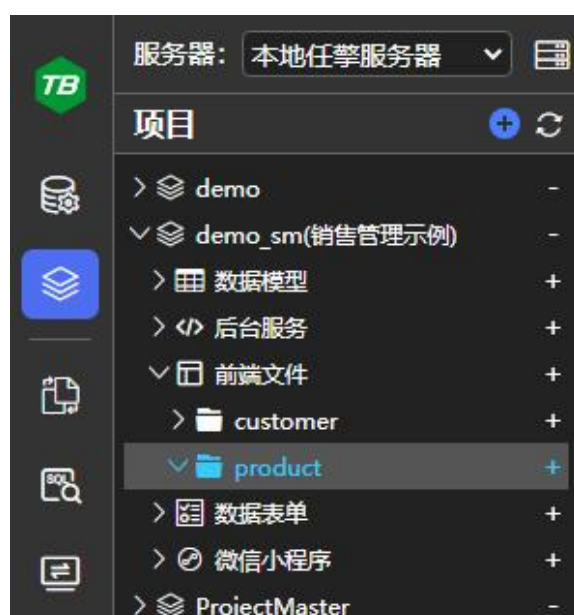
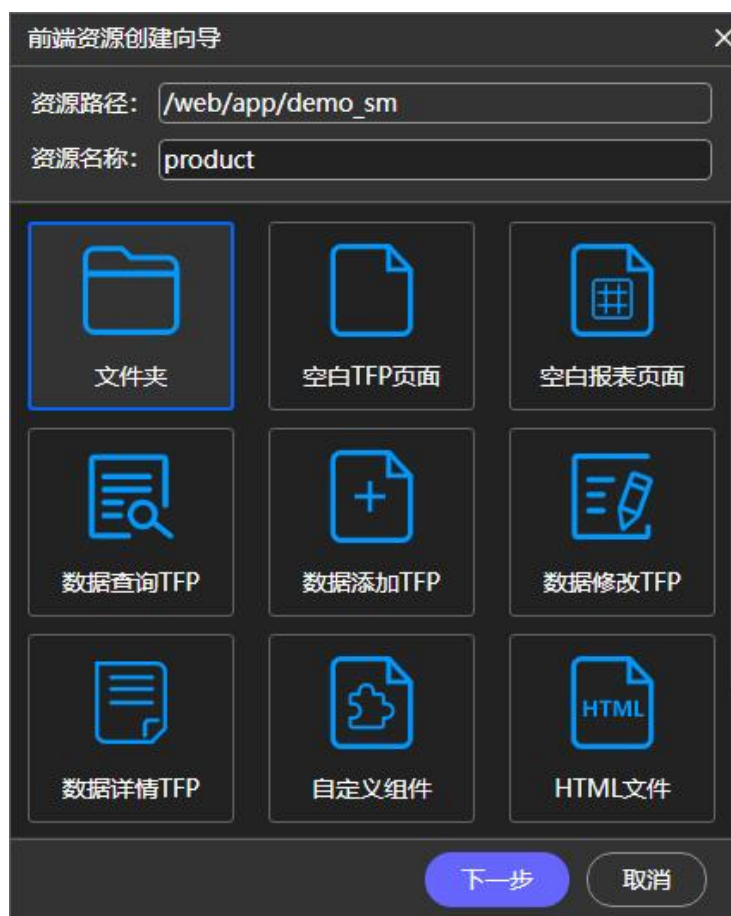
## 4.3 创建产品信息文件夹

点击销售管理示例项目“前端文件”右侧的加号按钮：



打开如下图所示的“前端资源创建向导”对话框，在该对话框中，选中“文件夹”，在资

源名称输入框中输入：product，然后点“下一步”按钮，就可以在销售管理示例项目的前端创建一个名为“product”的文件夹，以便存放所有产品管理有关的前端文件。

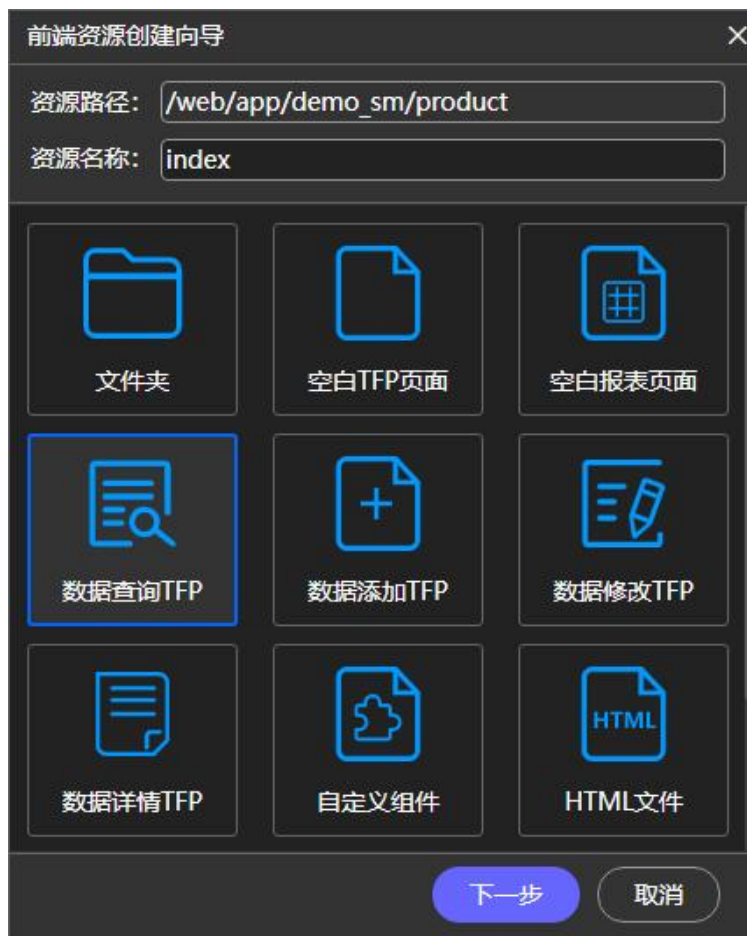




## 4.4 创建产品信息列表页面

### 4.4.1 新建产品信息列表页面

接下来，我们创建产品信息列表页面，在销售管理示例项目的前端文件里，点击 product 文件夹右侧的加号按钮，打开“前端资源创建向导”，资源类型选择“数据查询 TFP”：



然后点击“下一步”按钮，打开“创建数据查询 TFP”向导对话框，在此先设置页面基本信息，先选择前面创建的产品信息数据模型：demo\_sm\_product.tdm，然后设置页面标题：产品管理，然后选中“查询条件显示在页面左侧”，设置后的界面如下图：

创建数据查询TFP

第一步：设置基本信息

保存路径：

数据模型：

选择...

页面标题：

页面类型：

窗口大小：宽  高

终端类型：

背景颜色：

定位方式：

查询设置：☒ 启用翻页

每页最多显示

☐ 条记录

最多查询

☐ 条记录

☐ 去掉重复项

其他设置：☒ 显示添加按钮

添加页面文件名：

添加页面类型：

☒ 显示修改按钮

修改页面文件名：

修改页面类型：

☒ 显示删除按钮

☒ 查询条件显示在页面左侧（仅支持PC端）

数据查询服务：☒ 自动创建 ☐ 使用已有服务

响应参数名：

选择...

数据删除服务：☒ 自动创建 ☐ 使用已有服务

选择...

下一步

取消

然后点击下一步，选择和设置需要查询的字段，如下图所示：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段

添加关联字段

| <input checked="" type="checkbox"/> | 字段名称      | 字段备注   | 字段格式(双击编辑)                                   | 关联数据字典(双击选择) | 字段别名      |
|-------------------------------------|-----------|--------|----------------------------------------------|--------------|-----------|
| <input checked="" type="checkbox"/> | id        | 编号     | id                                           |              |           |
| <input checked="" type="checkbox"/> | code      | 编码     | code                                         |              |           |
| <input checked="" type="checkbox"/> | name      | 名称     | name                                         |              |           |
| <input type="checkbox"/>            | type_id   | 产品分类编号 | type_id                                      |              |           |
| <input type="checkbox"/>            | sort_num  | 分类序号   | sort_num                                     |              |           |
| <input checked="" type="checkbox"/> | model     | 规格型号   | model                                        |              |           |
| <input checked="" type="checkbox"/> | unit      | 计量单位   | unit                                         | unit         |           |
| <input checked="" type="checkbox"/> | price     | 单价     | price                                        |              |           |
| <input checked="" type="checkbox"/> | thumbnail | 缩略图代码  | thumbnail                                    |              |           |
| <input checked="" type="checkbox"/> | comment   | 备注     | comment                                      |              |           |
| <input checked="" type="checkbox"/> | status    | 状态     | status                                       |              |           |
| <input checked="" type="checkbox"/> | type_name | 分类名称   | (select name from demo_sm_product_type where |              | type_name |

上一步

下一步

取消

这里有几个特殊设置说明如下：

- **不需要查询的字段：**包括 type\_id（后面会关联查询分类名称，不需要显示分类编号）、sort\_num（用来排序，不需要显示）、deletor\_id、delete\_time（删除时生成）；
- **需要关联数据字典的字典：**unit，需要关联数据字典里的系统数据的计量单位；

- **需要关联查询的字段：** type\_name，需要关联产品分类表 demo\_sm\_product\_type 使用 type\_id 查询对应的分类名，点击上面的“添加关联字段”按钮，可以打开“添加关联字段”对话框，在此设置关联信息，具体设置如下图：

然后再点下一步，设置产品信息列表的显示格式，在此需要先删除编号字段，因为编号只是用来作为查询参数，不需要在列表里显示。然后其他需要特殊设置的字段及参数如下表：

| 列名   | 列类型 | 数据绑定格式                         | 文本显示格式 | 对齐方式 | 宽度  |
|------|-----|--------------------------------|--------|------|-----|
| 缩略图  | 图片  | /Download?fileCode={thumbnail} |        |      | 120 |
| 产品名称 |     |                                |        |      | 150 |
| 规格型号 |     |                                |        |      | 200 |
| 单位   |     |                                |        |      | 60  |

|    |  |  |    |     |     |
|----|--|--|----|-----|-----|
| 单价 |  |  | 金额 | 右对齐 | 80  |
| 备注 |  |  |    |     | 200 |
| 状态 |  |  |    |     | 60  |

其他字段默认即可，最终设置后的界面如下图所示：

创建数据查询TFP

第三步：设置显示格式

| 列名   | 列类型 | 数据绑定格式(js表达式)            | 文本显示格式 | 对齐方式 | 宽度  | 文字颜色 | 加粗 | 求和 |  |
|------|-----|--------------------------|--------|------|-----|------|----|----|--|
| 缩略图  | 图片  | /Download?fileCode={thun |        | 居中对齐 | 120 |      |    |    |  |
| 产品编码 | 文本  | {code}                   |        | 居中对齐 |     |      |    |    |  |
| 产品名称 | 文本  | {name}                   |        | 居中对齐 | 150 |      |    |    |  |
| 产品分类 | 文本  | {type_name}              |        | 居中对齐 |     |      |    |    |  |
| 规格型号 | 文本  | {model}                  |        | 居中对齐 | 200 |      |    |    |  |
| 单位   | 文本  | {unit}                   |        | 居中对齐 | 60  |      |    |    |  |
| 单价   | 文本  | {price}                  | 金额     | 右对齐  | 80  |      |    |    |  |
| 备注   | 文本  | {comment}                |        | 居中对齐 | 200 |      |    |    |  |
| 状态   | 文本  | {status}                 |        | 居中对齐 | 60  |      |    |    |  |

上一步 下一步 取消

然后点击下一步，设置动态查询参数，如下图所示：

创建数据查询TFP

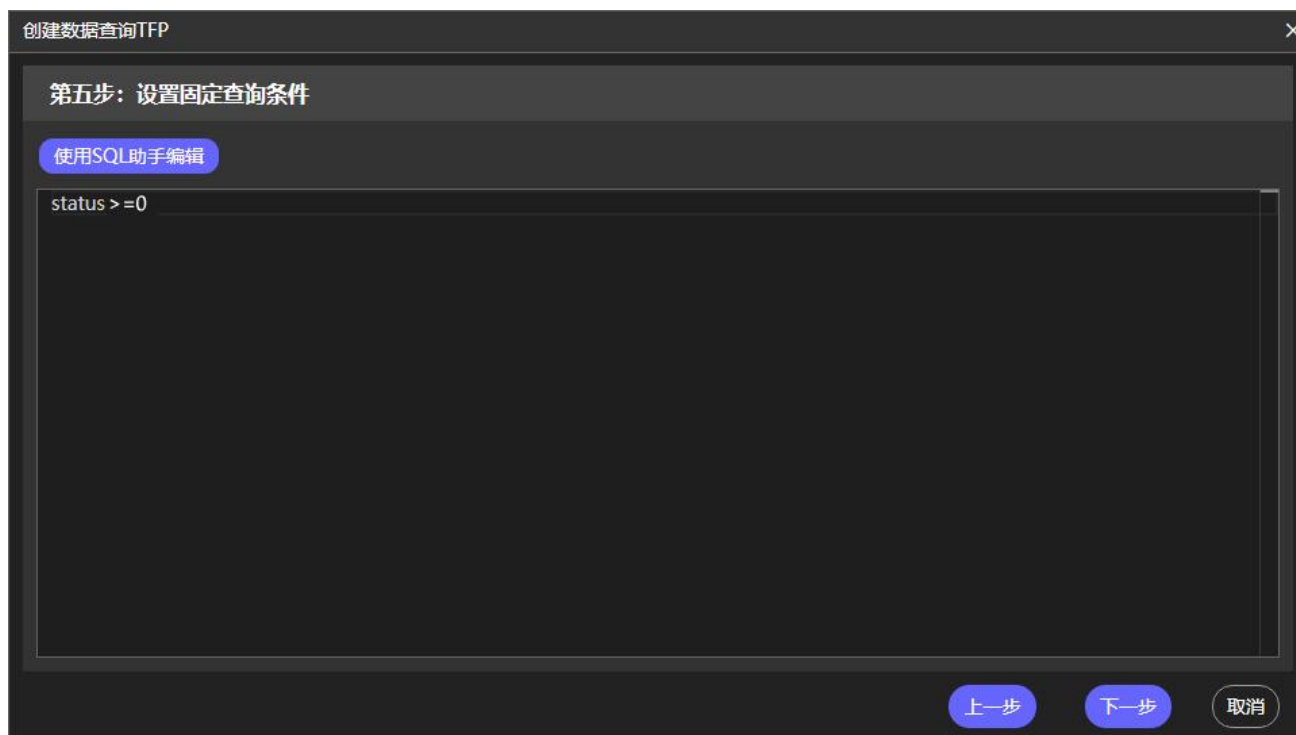
第四步：设置动态查询参数

查询字段: 查询条件:

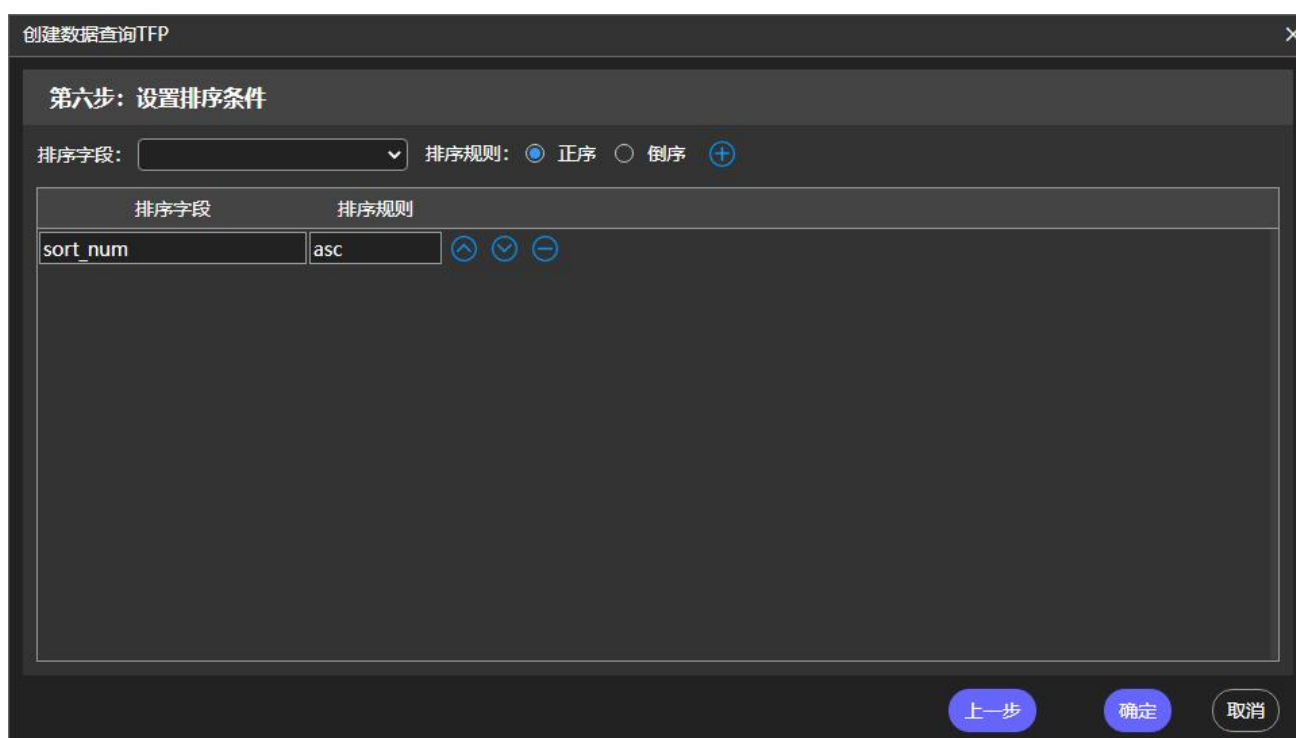
| 参数名称  | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)              | 默认值 | 数据字典(双击选择) | 可选值 |
|-------|------|-------|----|--------------------------|-----|------------|-----|
| code  | 编码   | 单行文本  |    | code like {%req.code%}   |     |            |     |
| name  | 名称   | 单行文本  |    | name like {%req.name%}   |     |            |     |
| model | 规格型号 | 单行文本  |    | model like {%req.model%} |     |            |     |

上一步 下一步 取消

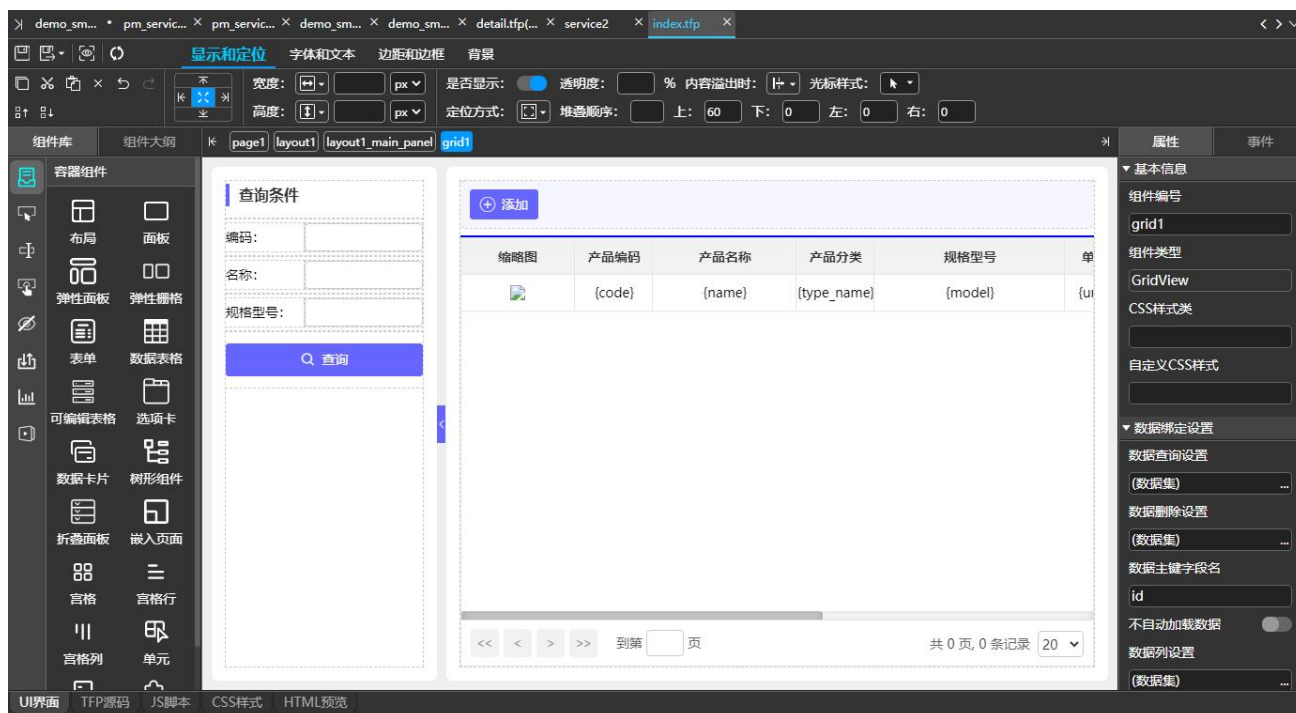
在此设置了三个动态查询参数，即可以按产品编码、名称和规格型号模糊查询，然后继续点击“下一步”按钮，显示如下所示界面，在此将固定查询条件设置为：status>=0，表示仅查询状态为正常或已禁用的产品信息。



然后继续点击“下一步”按钮，进入第六步，设置按 sort\_num 字段正序排列，如下图：



然后点“确定”按钮，即可完成产品信息列表页面的创建，并自动在 TaskBuilder 里打开，如下图所示：



#### 4.4.2 完善产品列表页面

我们准备在产品列表里显示产品的缩略图，但用向导创建的产品列表页面，数据表格组件的行高默认只有 40 像素，这显然是不够用的，需要调得更高一些，设置方法如下：

在产品信息列表页面里，选中数据表格组件 grid1，然后在右侧的属性设置里，将表体设置里的行高设置为：120，这样在产品信息列表里，就可以很好地显示缩略图了。

然后点击属性里的“数据列设置”，选中“缩略图”这一列，然后在右边的基本信息中，将“点击后执行的脚本”设置为：`tfp.openImage('/Download?fileCode={thumbnail}')`，表示点击后打开图片查看器查看大图，然后在中间的选项分类里选择“图片设置”，将图片高度和宽度都设置为 96 像素，然后点确定，保存设置即可。

另外，状态列建议设置显示条件，针对不同的状态显示不同的文字和颜色，参考客户列表。

由于现在没有添加任何产品数据，就先不预览测试了。



## 4.5 实现产品分类树形展示

### 4.5.1 树形组件简介

在我们开发的各种应用功能中，有不少数据是树形结构的，即数据之间有上下级关系，例如部门信息、应用分类、数据字典分类等等，这些数据可以分为多级，每一级我们称为“节点”，一个节点可以有多个子节点，但最多只能有一个父节点，一级节点我们称为“根节点”，即没有父节点的顶层节点。

在我们设计这类树形结构的数据模型时，至少需要设置三个字段：

- **节点编号**：一般用 id，目前仅支持整数类型，用来唯一识别某一个节点，值不能重复；
- **节点名称**：一般用 name，用来作为节点的文本内容显示在树形列表里，允许有重复值，例如在系统功能菜单里，多个大的功能模块里可能都有名为“参数设置”的子模块，它们虽然名称一样，但实际对应的页面地址是不一样的；
- **父节点编号**：一般用 parent\_id，用来存储当前节点的父节点的编号，通过该字段的值，就可以知道当前节点是另外哪个节点的子节点，对于一级节点（根节点），因为他们没有父节点，所以他们的父节点编号需要设置为一个固定的整数值，例如 0。

为了方便显示和管理树形结构的数据，TaskBuilder 提供了一个专门的“树形组件”，如果一个数据模型有了上面三个字段，并且里面的数据通过父节点编号形成了上下级关系，就可以使用树形组件通过数据查询设置关联该数据模型查询数据并采用树形结构展示。

树形组件默认只是展示数据，如果要编辑和删除数据，需要在该组件的属性中开启“显示节点编辑按钮”和“显示节点删除按钮”，这样在鼠标放到树形组件的节点上时，在节点右侧就会显示编辑和删除按钮，另外，如果想让用户可以选择节点，则需要在属性里开启“显示节点选择框”，默认是多选，也可以开启“允许多选”选项，支持多选。还有一些其他更细节的

属性设置，这里就不详细介绍了，可以参考树形组件的 API 文档。

另外，为了增强树形组件的功能，我们还提出了“叶结点”的概念，叶结点是指可以作为上面说的节点的子节点，但其自身不能再有子节点的末端节点，例如在组织结构树中，部门是节点，可以有多级部门，人员就是叶结点，人员下面不能再有其他子节点，还有系统文件夹和文件也是这种关系，文件夹可以有多层，但到文件就不能再展开了，也就是说，如果两种数据之间存在一对多的关联关系，且上层数据可以支持多级，下层数据只有一级，则可以采用“节点+叶结点”组合的形式按树形显示这两类数据。在树形组件的属性设置里，也提供了叶结点相关的各种属性设置，它们的功能基本与节点的一样，其中最重要的一个属性就是“叶父节点 ID 字段名”，用来设置叶节点的数据里，哪个字段存储的它所属的上级节点的 ID 值，例如在组织机构中，用户信息里，部门编号对应的字段名就是这里所说的“叶父节点 ID 字段名”。

我们接下来要开发的产品分类信息就是典型的树形结构，为了实现产品分类信息的查看和管理，我们可以通过在产品信息列表页面左侧放置一个树形组件并设置相关参数来实现，具体的操作方法如下：

#### 4.5.2 调整查询条件显示位置

我们前面用向导创建产品信息列表页面时，设置了三个查询条件，并显示在了页面左侧，现在，我们想在页面左侧显示产品分类的树形组件，以便对产品分类进行管理，也方便按产品分类查询产品信息，这样就需要把前面创建的这几个查询条件的文本标签和输入项挪到产品列表上面的面板中，放到添加按钮后面，下面以产品编码为例，介绍一下具体的操作方法。

在产品信息列表页面中，用鼠标左键按住左侧面板里的“编码：”文本标签，然后拖拽到产品列表上部的面板组件中，然后松开鼠标，即可以将该组件从左侧面板挪到列表上部的面板组件中，如下图所示：



挪位置之前



挪位置之后

这时我们可以看到，“编码：”文本标签与前面的添加按钮之间没有间距，挨得太近了，需要设置一下间距，设置方法如下：

选中这个文本标签组件，然后点击前端页面设计器上部样式设置栏里的“边距和边框”，然后设置左外边距为 15 像素，按回车，这样看着就比较美观了，如下图所示：



然后, 我们继续将编码查询条件的单行文本输入框从左侧面板拖拽到右侧产品信息列表上部的面板中, 拖拽完后, 发现该输入项的宽度太宽了, 这是因为它的宽度是 100%, 需要修改成具体的像素值, 修改方法如下:

选中该单行文本输入框, 然后点击前端页面设计器上部样式设置栏里的“显示和定位”, 将宽度的单位改成“px”, 如下图所示:

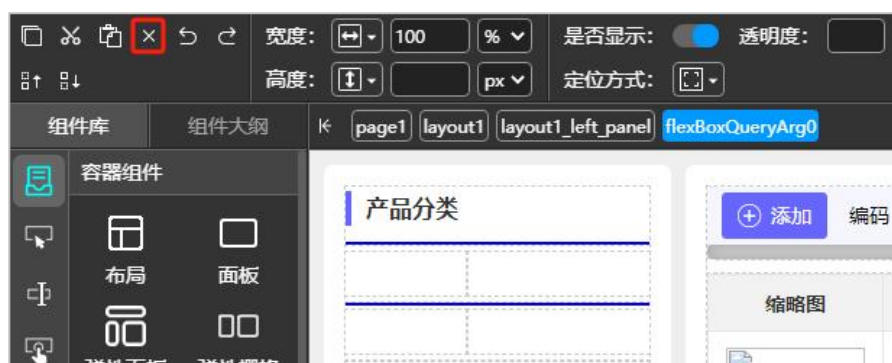


这样就显示正常了, 然后我们依次把名称和规格型号这两个查询条件的文本标签和输入项, 以及查询按钮都拖拽到产品信息列表上部的面板中, 并设置好组件的宽度和间距, 最终效果如下图:



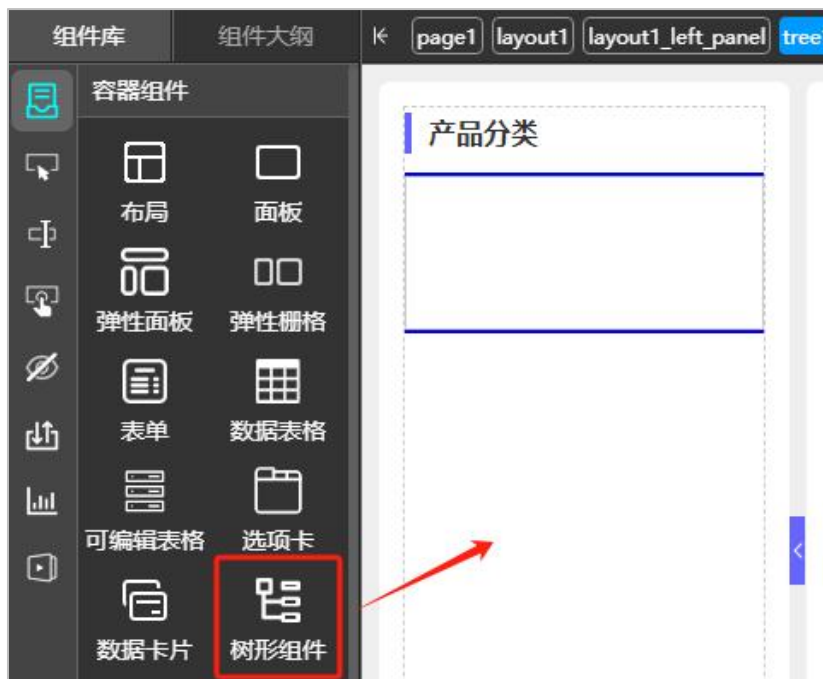
接下来, 在页面的左侧面板中, 选中最上面的“查询条件”文本标签, 在属性设置里面, 将“组件文本”改成: 产品分类。

最后, 选中原来查询条件输入项和查询按钮所在的各个弹性栅格组件, 点击工具栏上的“删除”按钮, 把这些弹性栅格组件都删除, 就可以为产品分类的树形组件清理出足够的空间。



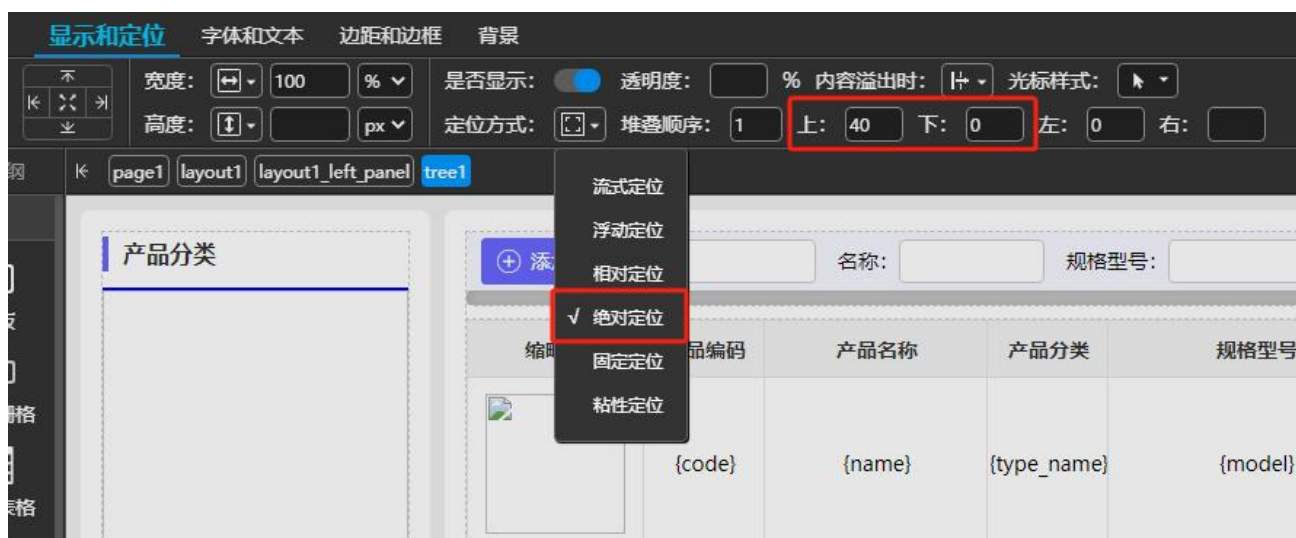
### 4.5.3 添加树形组件

在前端页面设计器组件库的容器组件中，用鼠标左键按住“树形组件”，然后拖拽到产品信息列表页面左侧的面板中，松开鼠标，即可在此新建一个树形组件，如下图所示。



### 4.5.4 设置树形组件定位方式

新建的树形组件，高度默认为 90 像素，这肯定不够，需要设置为根据页面高度动态变化，设置方法如下：

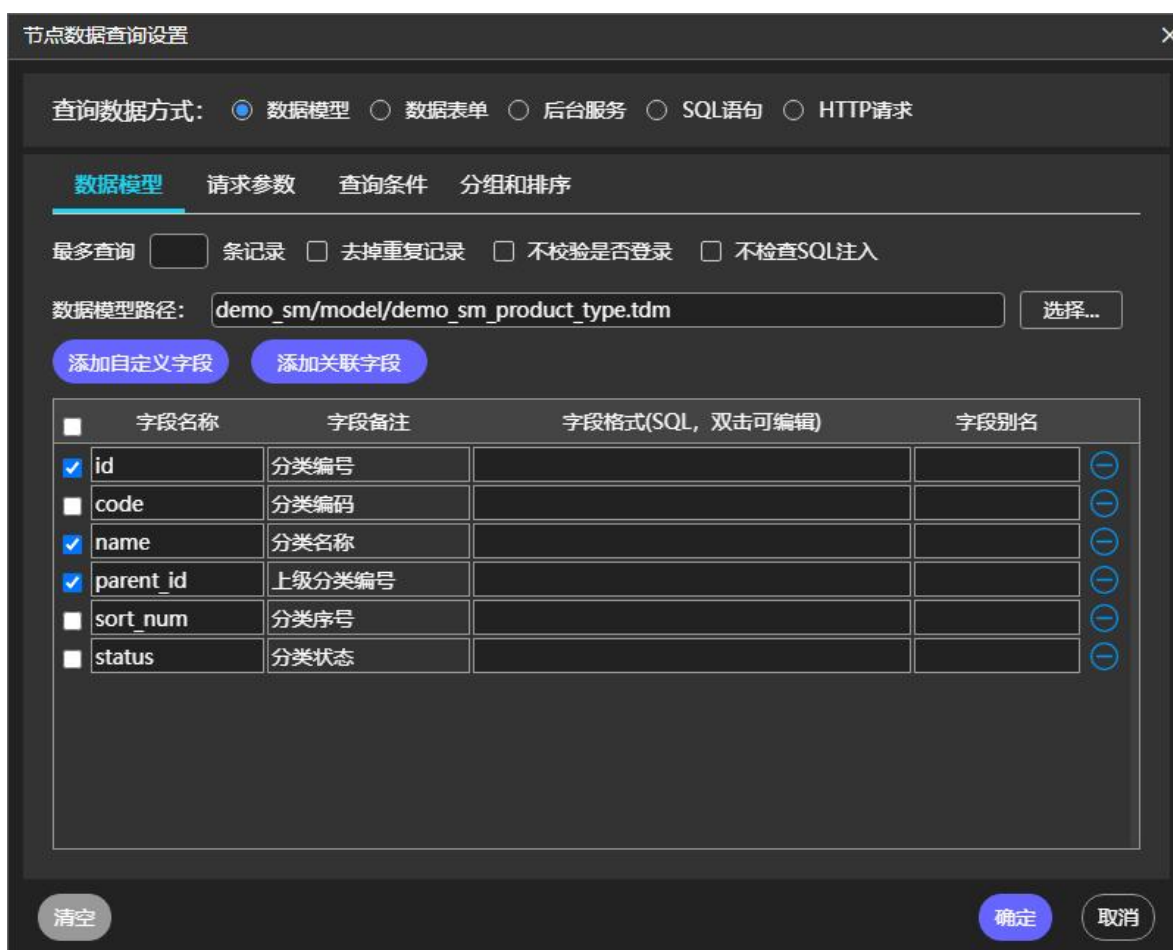


选中树形组件, 在上面的样式设置栏的显示和定位里, 点击定位方式右边的列表, 选择“绝对定位”, 然后设置相对父组件边框的顶部距离为 40, 底部距离为 0, 这样设置好后, 就可以看到, 在页面左侧面板中, 除了顶部的产品分类标题栏, 剩下的区域都被树形组件占据了。

另外, 在上面样式设置栏里的边距和边框中, 将树形组件的边框设置为 0, 会更加美观。

#### 4.5.5 设置树形组件数据查询有关参数

然后点击树形组件属性里的“节点数据查询设置”, 打开下图所示对话框:



节点数据查询设置

查询数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据模型** 请求参数 查询条件 分组和排序

最多查询  条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入

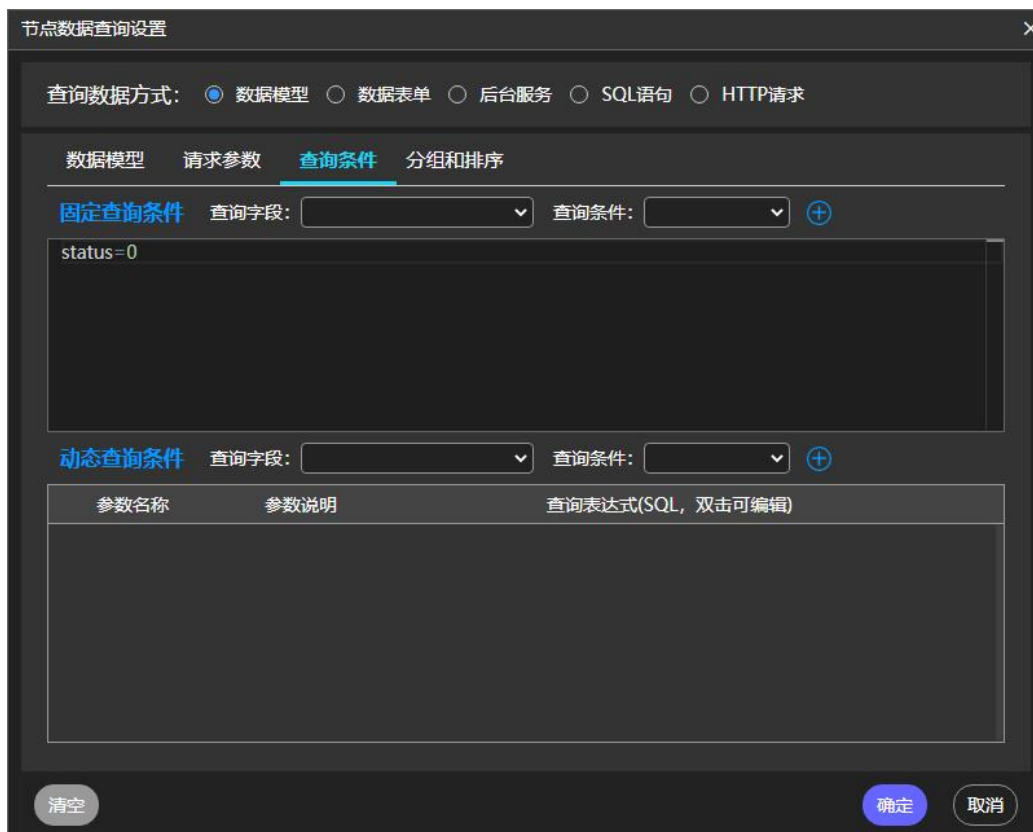
数据模型路径:

| ■                                   | 字段名称      | 字段备注   | 字段格式(SQL, 双击可编辑) | 字段别名 |   |
|-------------------------------------|-----------|--------|------------------|------|---|
| <input checked="" type="checkbox"/> | id        | 分类编号   |                  |      | ⊖ |
| <input type="checkbox"/>            | code      | 分类编码   |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | name      | 分类名称   |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | parent_id | 上级分类编号 |                  |      | ⊖ |
| <input type="checkbox"/>            | sort_num  | 分类序号   |                  |      | ⊖ |
| <input type="checkbox"/>            | status    | 分类状态   |                  |      | ⊖ |

在该对话框中, 先选择好数据模型: demo\_sm\_product\_type.tdm, 再选中 id、name 和 parent\_id 这三个字段。

然后点击上面的“查询条件”选项卡, 将固定查询条件设置为: status=0。





节点数据查询设置

查询数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 查询条件 分组和排序

**固定查询条件** 查询字段:  查询条件:  +

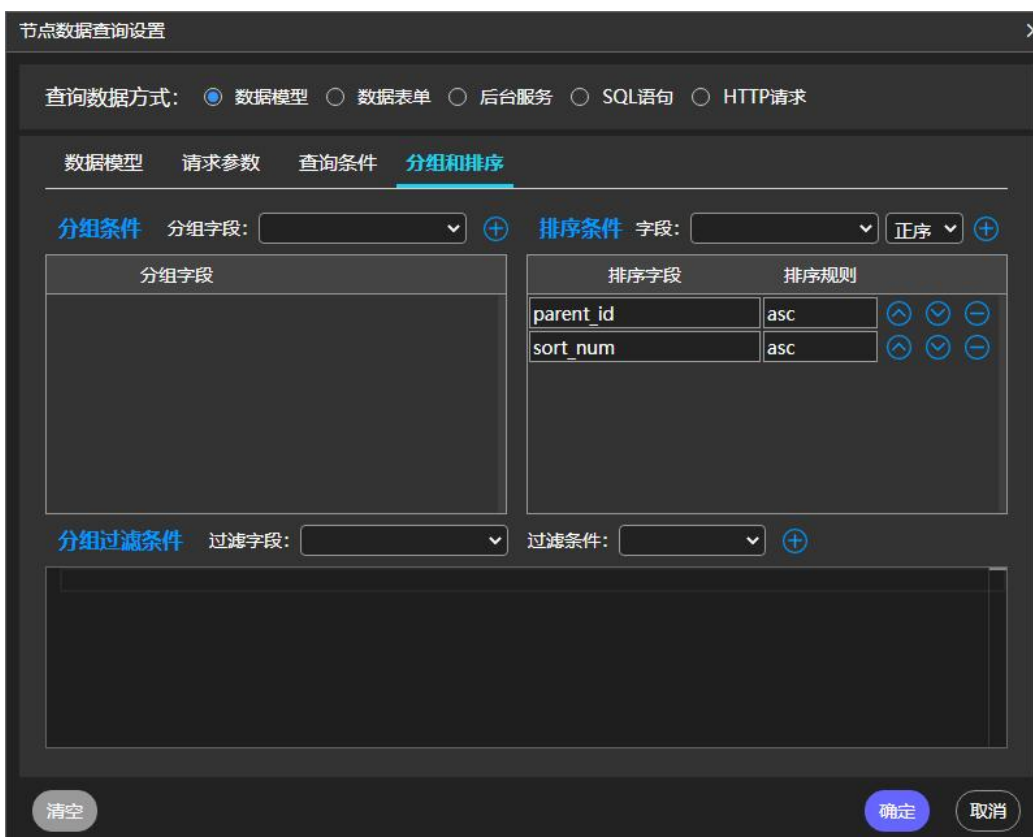
status=0

**动态查询条件** 查询字段:  查询条件:  +

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

清空 确定 取消

然后点击上面的“分组和排序”选项卡，按照下图所示设置排序条件：



节点数据查询设置

查询数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 查询条件 分组和排序

**分组条件** 分组字段:  +

**排序条件** 字段:  排序规则: 正序 +

| 排序字段      | 排序规则 |
|-----------|------|
| parent_id | asc  |
| sort_num  | asc  |

**分组过滤条件** 过滤字段:  过滤条件:  +

清空 确定 取消

然后点确定，完成树形组件节点数据查询的设置。

在页面实际运行时，树形组件会根据以下几个属性的设置进行数据绑定：

- **根节点父节点 ID**：用来设置一级节点的父节点 ID 值，默认为 0；
- **节点 ID 字段名**：用来设置树形组件每个节点 ID 值对应的字段名，默认为：id；
- **节点名称字段名**：用来设置树形组件每个节点显示的文本对应的字段名，默认为：name；
- **父节点 ID 字段名**：用来设置父节点 ID 值对应的字段名，默认为：parent\_id；

以上这几个属性使用默认值即可，不需要修改，因为这里默认设置的字段名与产品分类数据模型里的字段名是一样的。

到此，就完成了产品分类树形显示的设置，但现在还没有添加数据，就先不预览测试了。

## 4.6 实现添加产品分类的功能

### 4.6.1 对话框简介

由于产品分类信息比较少，如果用普通页面或抽屉页显示，页面内会出现很多空白区域，浪费空间，也不够美观，所以准备采用对话框形式显示产品分类的添加和修改页面。

在对话框弹出时，默认会显示在整个浏览器窗口的正中央，底下会显示一个透明的遮罩层，将浏览器内当前窗口的其他内容都遮挡住，虽然还能看到这些内容，但无法点击或编辑，必须在关闭对话框后，才能继续操作。

需要注意的是，在前端页面设计器里打开对话框页面，在属性里设置的高度和宽度仅供设计时参考，对话框显示时的实际高度和宽度是在“打开对话框”语句里设置的。

在 TaskBuilder 的前端页面设计器的 JS 脚本编辑界面中，在左侧的语句库里点击“对话框操作”这个语句分类图标，就可以看到，里面提供了很多常用的对话框操作语句，包括打开对话框、关闭当前对话框、获得当前对话框等，开发者可以根据需要拖拽这些语句到右边的编辑区中进行细节设置。



下面在介绍实现添加和修改产品分类信息的功能时会讲解对话框的具体使用。

### 4.6.2 设置添加产品分类按钮

在前端页面设计器里，从左侧组件库的只读组件里，按住按钮组件拖拽到页面左侧面板产品分类标签的右边，并设置以下属性：

- **按钮文本**：设置为一个 + 号；
- **按钮类型**：设置为主要按钮；
- **按钮风格**：设置为椭圆形；
- **文本高度**：设置为 22；

然后在上面的样式设置栏里，设置以下样式：

- **宽度**：设置为 24px；
- **高度**：设置为 24px；
- **外边距的上边距**：设置为 3；

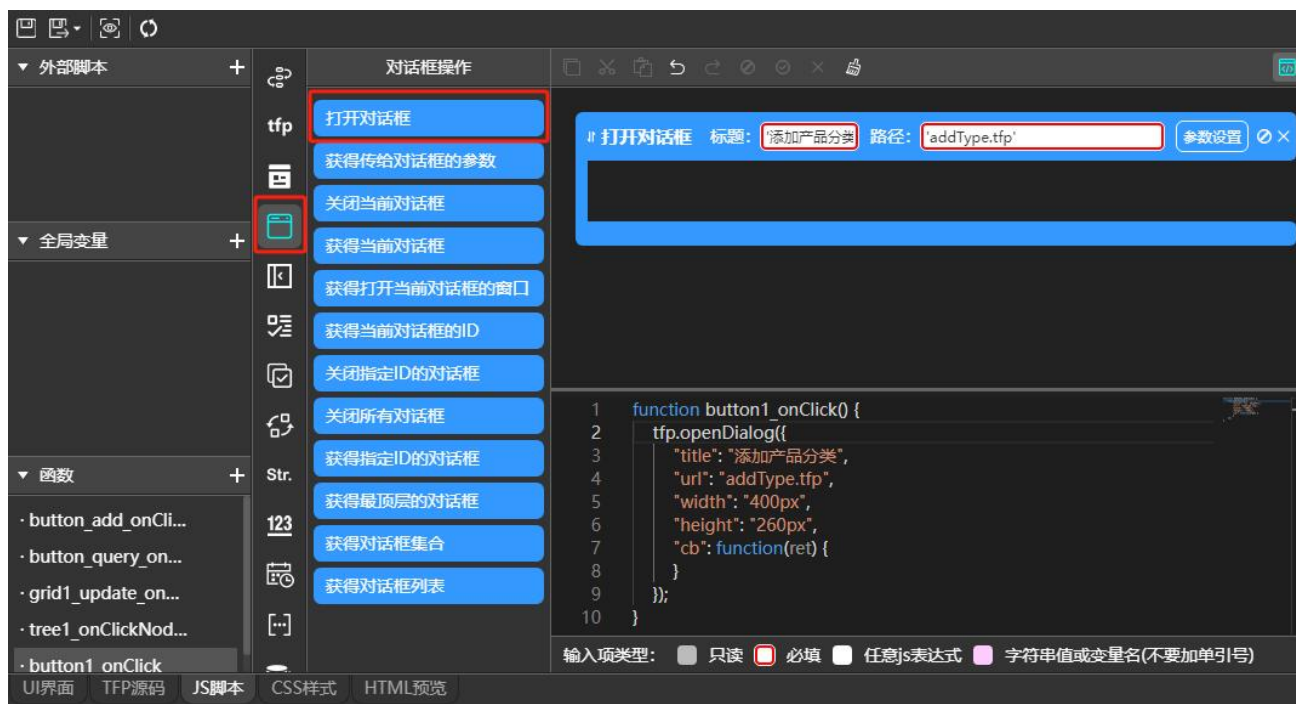
- **外边距的左边距**：设置为 100；

最终的效果如下图所示：



选中该按钮，在右侧的事件设置面板里，双击“点击时”事件输入框，打开事件函数设置界面，在此，先从左侧语句库的对话框操作中，拖拽一个“打开对话框”语句到设计区，然后设置以下参数：

- **标题**：设置为 '添加产品分类'；
- **路径**：设置为 'addType.tfp'；
- **宽度**：设置为 '400px'；
- **高度**：设置为 '260px'；



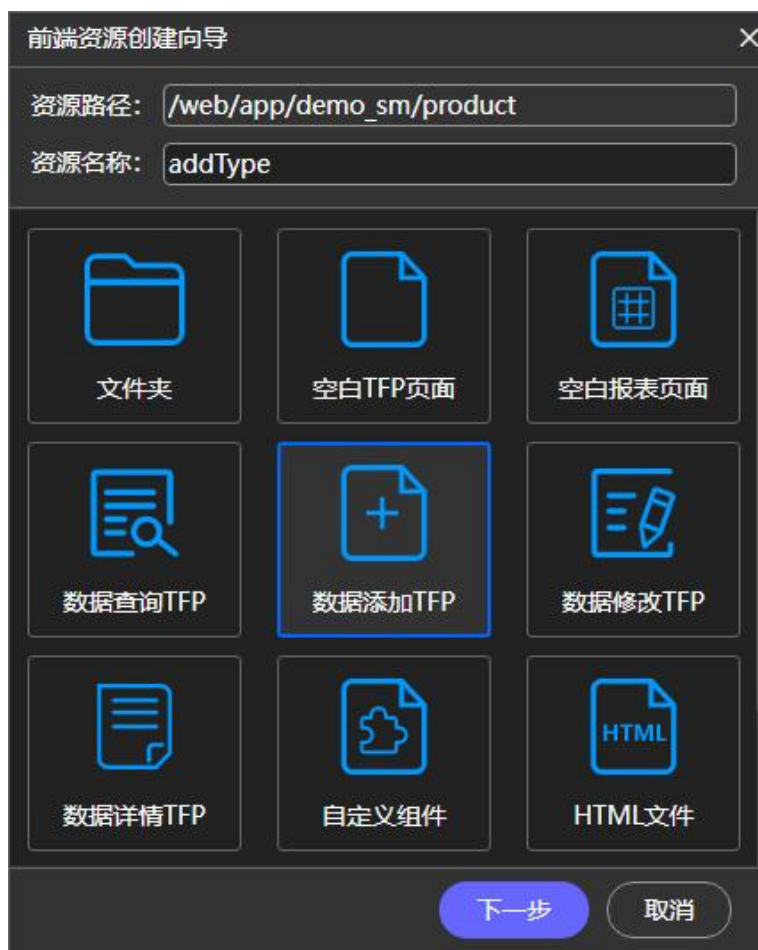
### 4.6.3 创建添加产品分类的对话框

前面设置好了添加产品分类的按钮，在该按钮的点击事件中，会以对话框形式打开添加产品分类信息的页面，采取对话框形式，是因为产品分类的信息比较少，如果用普通页面或抽屉

页，会造成页面内有大量空白区域，不太美观。

接下来我们带领大家创建添加产品分类的对话框页面，具体操作如下：

在销售管理示例项目的前端文件中，点击 product 文件夹右侧的加号，打开“前端资源创建向导”对话框，资源类型选择“数据添加 TFP”，资源名称设置为：addType：



然后点下一步，打开“创建数据添加 TFP”对话框，在该对话框中，设置以下参数：

- **数据模型：**选择 demo\_sm\_product\_type.tdm；
- **页面标题：**设置为添加产品分类；
- **页面类型：**设置为对话框；
- **窗口宽度：**设置为 400；
- **窗口高度：**设置为 260，当修改文本标签显示位置、每行显示输入项数量和前端要显示的字段后，会自动重新计算窗口的高度；

- **输入项文本标签显示位置**：选择显示在输入项左侧；
- **每行显示输入项数量**：设置为 1；

然后在下面的前端页面设置字段里选中以下字段，并按照下表设置相关参数：

| 字段        | 输入项名称 | 输入项类型   | 必填 | 默认值 |
|-----------|-------|---------|----|-----|
| code      | 分类编码  | 单行文本输入框 | 是  |     |
| name      | 分类名称  | 单行文本输入框 | 是  |     |
| parent_id | 上级分类  | 下拉列表    | 否  | 0   |
| sort_num  | 分类序号  | 单行文本输入框 | 是  |     |

其中，parent\_id 的默认值为 0，表示如果不选择上级分类，则表示是一级分类。

最终设置好的界面如下图所示：

创建数据添加TFP

保存路径: /web/app/demo\_sm/product/addType.tfp 数据模型: demo\_sm/model/demo\_sm\_product\_type.tdm 选择...

页面标题: 添加产品分类 页面类型: 对话框 窗口大小: 宽 400 高 260

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☐ 输入项上部 ☒ 输入项左侧 每行显示输入项数量: 1 仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务 选择...

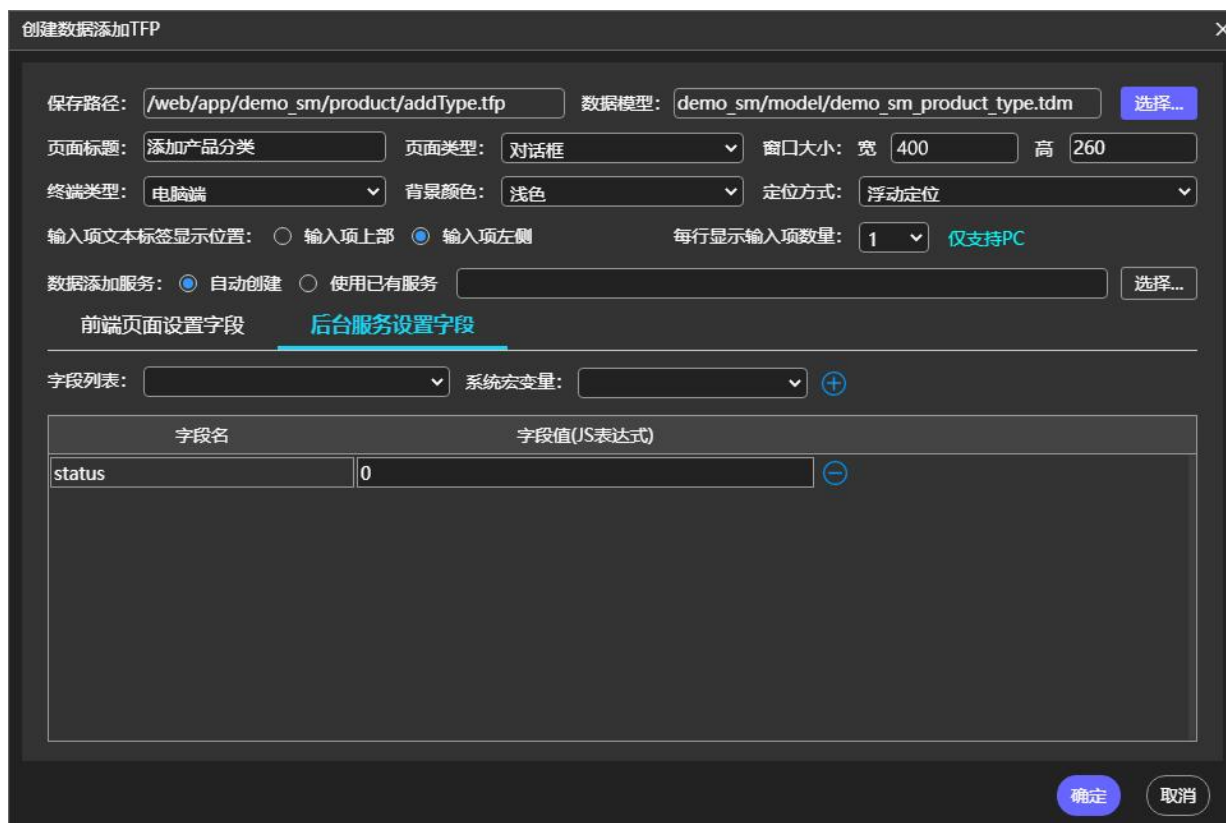
**前端页面设置字段** 后台服务设置字段

| ■                                   | 字段        | 输入项名称 | 输入项类型 | 列数 | 必填                                  | 可选值 | 数据字典(双击选择) | 默认值 |
|-------------------------------------|-----------|-------|-------|----|-------------------------------------|-----|------------|-----|
| <input checked="" type="checkbox"/> | code      | 分类编码  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |            |     |
| <input checked="" type="checkbox"/> | name      | 分类名称  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |            |     |
| <input checked="" type="checkbox"/> | parent_id | 上级分类  | 下拉列表  | 1  | <input type="checkbox"/>            |     |            | 0   |
| <input checked="" type="checkbox"/> | sort_num  | 分类序号  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |     |            |     |
| <input type="checkbox"/>            | status    | 分类状态  | 单行输入框 | 1  | <input type="checkbox"/>            |     |            |     |

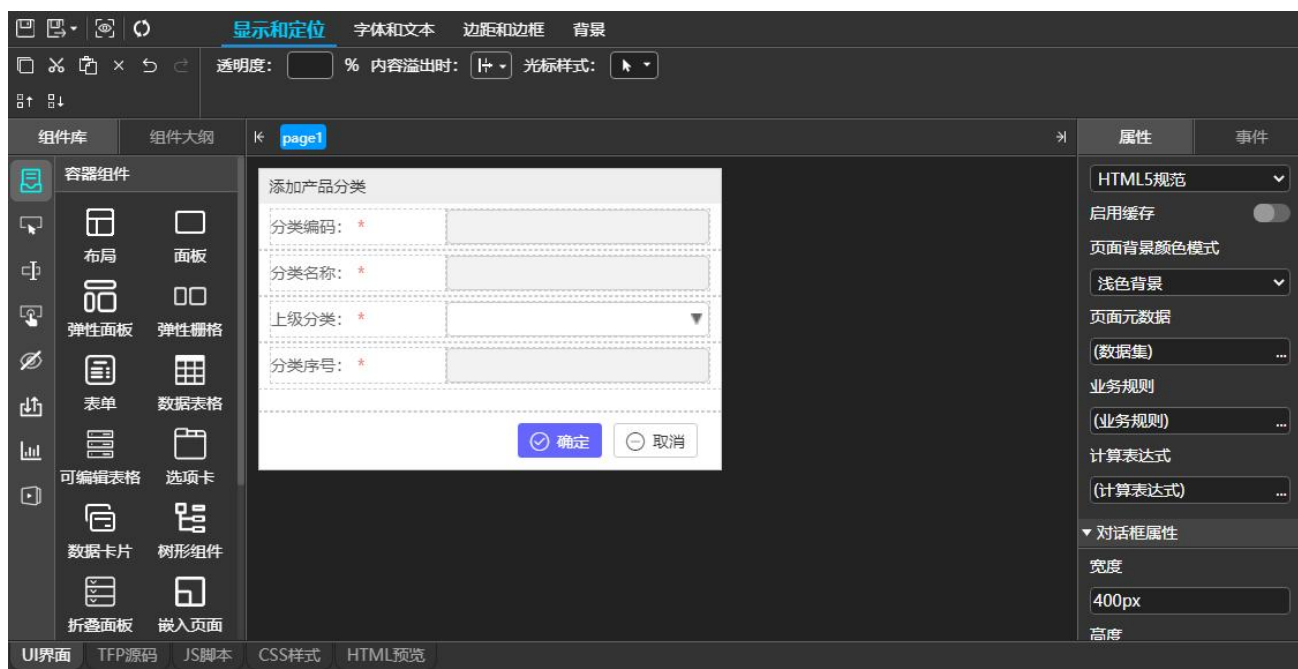
确定 取消

然后点击“后台服务设置字段”选项卡，添加 status 字段，并将值设置为 0，表示新添加的产品分类是正常有效的，设置好的界面如下图所示：





最后点击确定按钮，即可完成添加产品分类信息页面的创建，并会自动在 TaskBuilder 里打开，界面如下所示：



上面用向导创建的添加产品分类信息对话框存在两个问题：

- (1) 上级分类下拉列表没有查询产品分类数据，无法选择；

(2) 分类序号需要手动输入，不能在现有序号基础上自动增长；

下面，我们带领大家完善这两个问题。

#### 4.6.4 设置上级分类下拉列表的数据查询参数

在上面创建的添加产品分类信息对话框中，选中上级分类的下拉列表组件（组件编号：parent\_id），然后点击属性里的“数据查询设置”，打开如下对话框：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据模型** 请求参数 查询条件 分组和排序

最多查询  条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

| <input type="checkbox"/>            | 字段名称      | 字段备注   | 字段格式(SQL, 双击可编辑) | 字段别名 |                                  |
|-------------------------------------|-----------|--------|------------------|------|----------------------------------|
| <input checked="" type="checkbox"/> | id        | 分类编号   |                  |      | <input type="button" value="⊖"/> |
| <input type="checkbox"/>            | code      | 分类编码   |                  |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | name      | 分类名称   |                  |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | parent_id | 上级分类编号 |                  |      | <input type="button" value="⊖"/> |
| <input type="checkbox"/>            | sort_num  | 分类序号   |                  |      | <input type="button" value="⊖"/> |
| <input type="checkbox"/>            | status    | 分类状态   |                  |      | <input type="button" value="⊖"/> |

在该对话框中，先选择数据模型：demo\_sm\_product\_type.tdm，然后在下面的字段列表中选中 id、name 和 parent\_id 这三个字段，然后点击上面的“查询条件”选项卡，在里面设置查询条件为：status=0，表示只查询有效的产品分类，如下图所示：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '查询条件' (Query Conditions) tab selected. The '查询数据方式' (Query Data Method) is set to '数据模型' (Data Model). Under '固定查询条件' (Fixed Query Conditions), the '查询字段' (Query Field) is empty and the '查询条件' (Query Condition) is 'status=0'. Under '动态查询条件' (Dynamic Query Conditions), the '查询字段' (Query Field) is empty and the '查询条件' (Query Condition) is empty. Below these is a table for '查询表达式(SQL, 双击可编辑)' (Query Expression (SQL, double-click to edit)).

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

Buttons at the bottom: 清空 (Clear), 确定 (Confirm), 取消 (Cancel).

然后再点击上面的“分组和排序”，在里面添加两个排序条件，即按 parent\_id（上级分类编号）和 sort\_num（分类序号）正序排列，如下图所示：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '分组和排序' (Grouping and Sorting) tab selected. The '查询数据方式' (Query Data Method) is set to '数据模型' (Data Model). Under '分组条件' (Grouping Conditions), the '分组字段' (Grouping Field) is empty. Under '排序条件' (Sorting Conditions), the '字段' (Field) is empty and the '排序规则' (Sorting Rule) is '正序' (Ascending). Below these is a table for '排序字段' (Sorting Field) and '排序规则' (Sorting Rule).

| 排序字段      | 排序规则 |
|-----------|------|
| parent_id | asc  |
| sort_num  | asc  |

Buttons at the bottom: 清空 (Clear), 确定 (Confirm), 取消 (Cancel).

然后点确定按钮，即可完成上级分类下拉列表组件的数据查询设置。

然后在该组件的属性里，继续设置以下属性：

- **选项值字段**：设置为 id；
- **选项文本字段**：设置为 name；
- **显示树形结构**：开启该选项；
- **父节点主键字段**：设置为 parent\_id；
- **根节点主键值**：设置为 0；

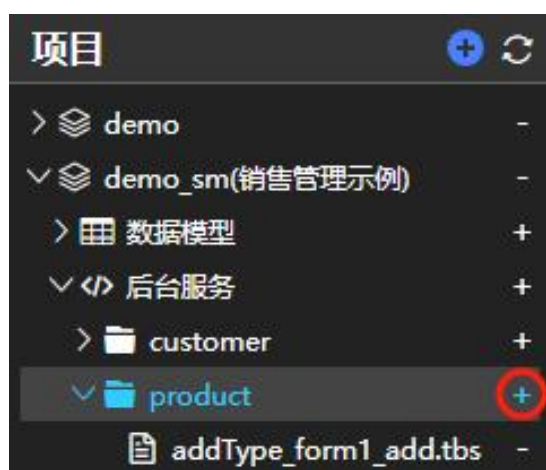
以上信息都设置好后，打开添加产品分类信息对话框时，就会自动查询已有的产品分类信息，并显示在该列表内。

## 4.6.5 自动生成分类序号

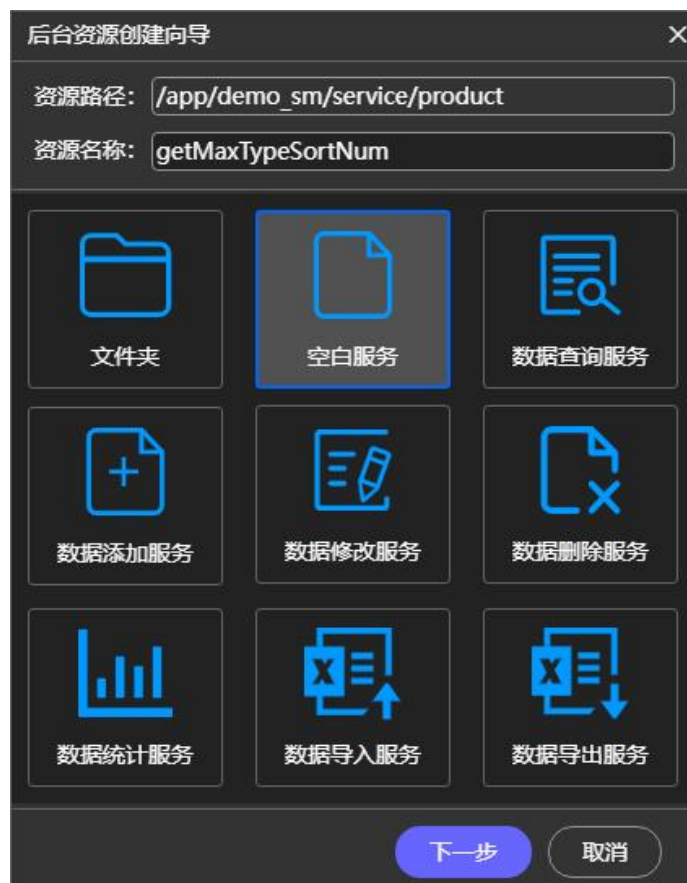
为了方便用户在添加产品分类时，参考已有的分类序号设置新建分类的序号，可以按照以下方法进行设置：

### 4.6.5.1 创建生成最大分类序号的后台服务

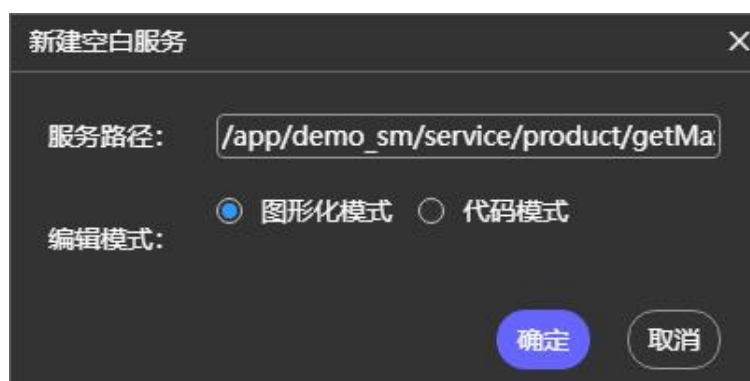
在销售管理示例项目的后台服务中，点击 product 文件夹右侧的加号按钮：



会弹出如下所示的“后台资源创建向导”对话框：



在该对话框中选择“空白服务”，资源名称设置为：getMaxTypeSortNum，然后点下一步，弹出“新建空白服务”对话框，如下图所示：



在此直接点击确定按钮，即可创建一个空白的后台服务，然后按下述步骤进行设置：

- (1) 先从基本语句中拖拽一个变量声明语句，将名称设置为 parentId，初始值设置为 0；
- (2) 然后再从基本语句中拖拽一个判断语句，将判断条件设置为：req.parent\_id，表示如果客户端发送了上级分类编号，则将该编号赋值给上面的 parentId 变量；

(3) 然后从数据库操作中拖拽一个数据查询操作，并点击该语句的设置按钮，设置相关参数，首先是基本信息设置，在此选择数据模型：demo\_sm\_product\_type.tdm，如下图所示：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '基本信息' (Basic Information) tab selected. The '数据模型' (Data Model) is set to 'demo\_sm/model/demo\_sm\_product\_type.tdm'. The '翻页设置' (Page Settings) section has '启用翻页' (Enable Paging) checked and '每页最多显示' (Maximum records per page) set to 20. The '查询数量' (Query Quantity) is set to '最多查询' (Maximum query) with a value of 1. The '重复设置' (Repeat Settings) section has '去掉重复记录' (Remove duplicate records) unchecked. The '查询结果参数名称' (Query result parameter name) is set to 'rows'. At the bottom right are '确定' (Confirm) and '取消' (Cancel) buttons.

然后点击查询字段，选中 sort\_num 字段，并设置其字段格式为：ifnull(max(sort\_num), 0)+1，表示查询这个字段的最大值，如果没有查到，则设置为 0，然后再加上 1，设置界面如下：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '查询字段' (Query Fields) tab selected. There are two buttons at the top: '添加自定义字段' (Add Custom Field) and '添加关联字段' (Add Associated Field). Below is a table with columns: '字段名称' (Field Name), '字段备注' (Field Remark), '字段格式(SQL, 双击可编辑)' (Field Format (SQL, double-click to edit)), '字段别名' (Field Alias), and '汇总' (Summary). The table contains the following data:

| 字段名称      | 字段备注   | 字段格式(SQL, 双击可编辑)           | 字段别名     | 汇总 |
|-----------|--------|----------------------------|----------|----|
| id        | 分类编号   |                            |          | ▼  |
| code      | 分类编码   |                            |          | ▼  |
| name      | 分类名称   |                            |          | ▼  |
| parent_id | 上级分类编号 |                            |          | ▼  |
| sort_num  | 分类序号   | ifnull(max(sort_num), 0)+1 | sort_num | ▼  |
| status    | 分类状态   |                            |          | ▼  |

At the bottom right are '确定' (Confirm) and '取消' (Cancel) buttons.



然后点击固定查询条件，在此将查询条件设置为：parent\_id={parentId}，如下图所示：



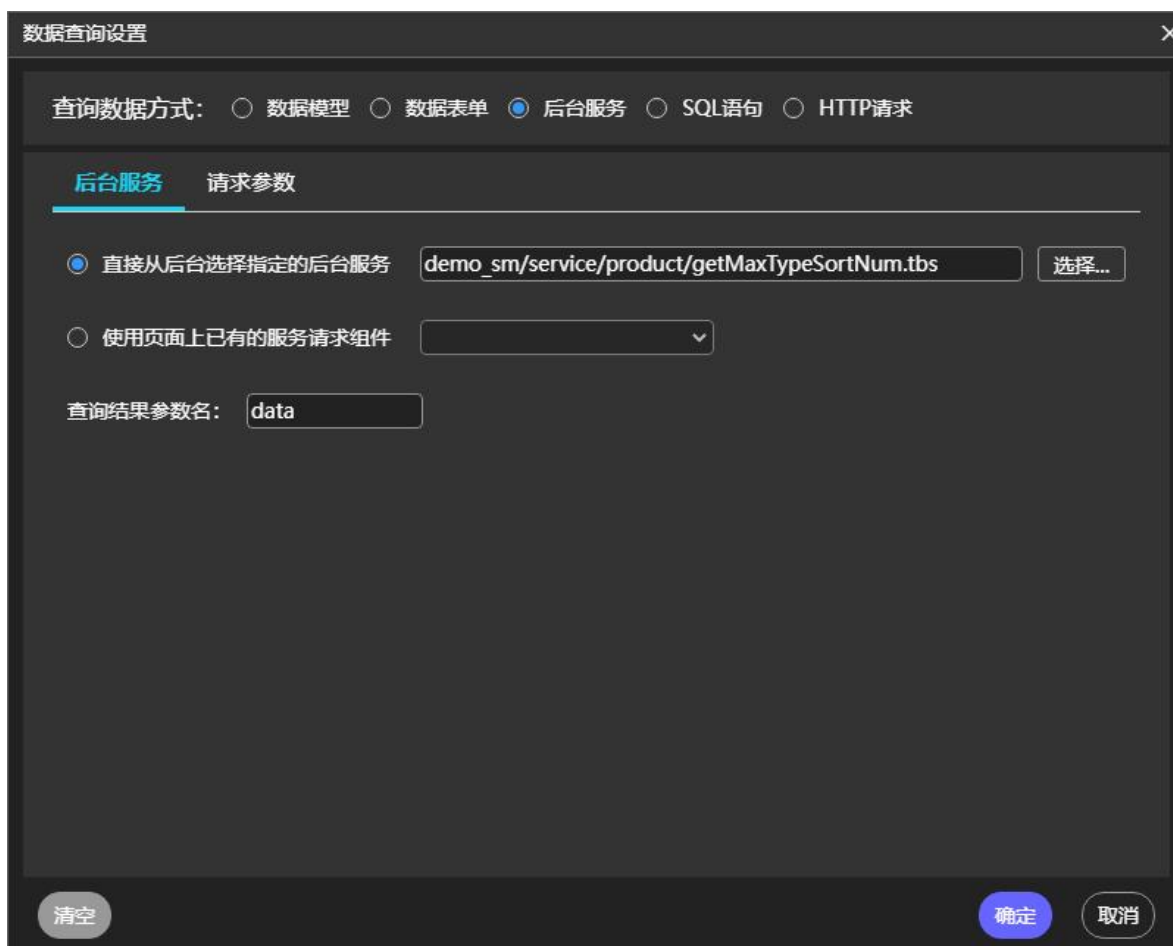
- (4) 然后从基本语句中拖拽一条变量赋值语句到数据查询语句中，将查询结果中的第一条记录 rows[0] 保存到服务器响应对象 res 的 data 属性中；
- (5) 最后终止服务，将 res 里的 data 数据返回给客户端。

最后设置完的后台服务如下图所示：



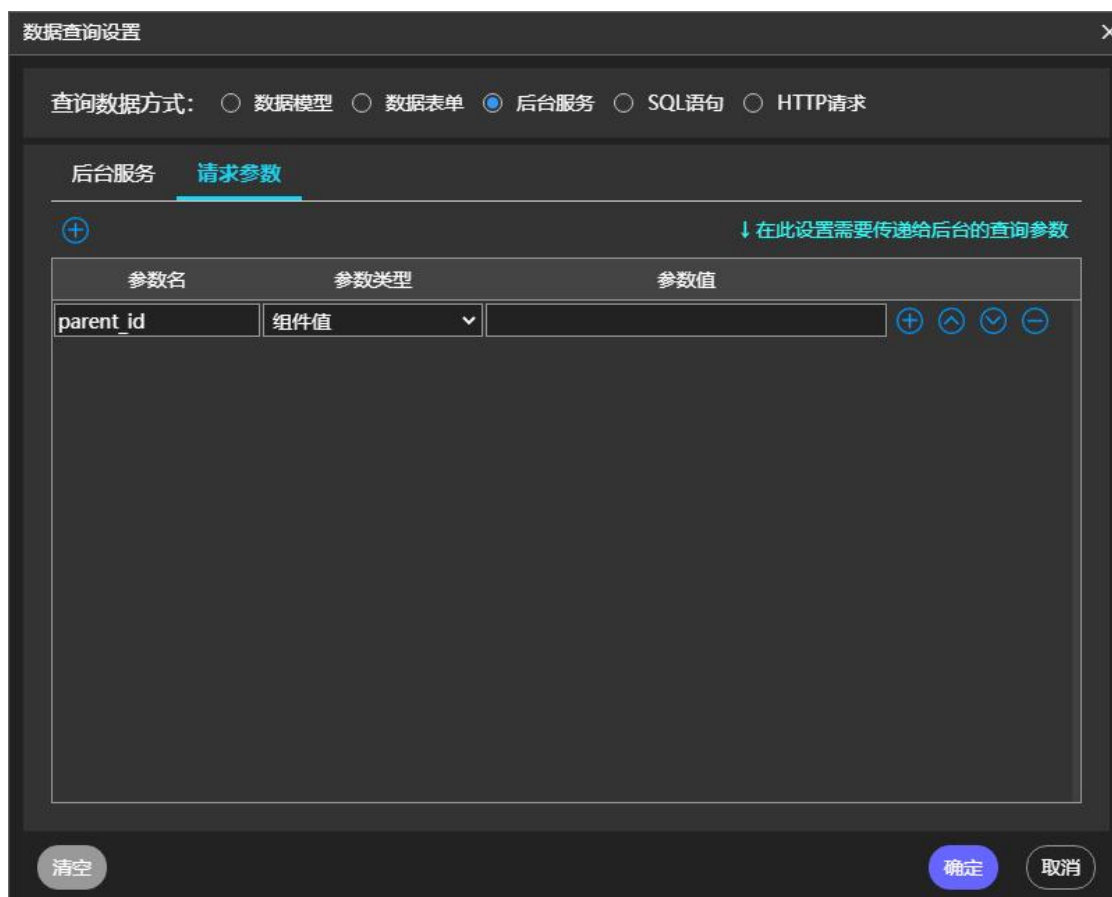
#### 4.6.5.2 设置表单的数据查询参数

在添加产品分类对话框里，选中表单组件 form1，点击属性里的“数据查询设置”，打开如下图所示对话框，在此将对话框顶部的查询数据方式设置为“后台服务”，然后选择上面创建的这个后台服务：demo\_sm/service/product/getMaxTypeSortNum.tbs，



然后点击请求参数，添加一个名为 parent\_id 的参数，参数类型设置为组件值，表示在查询目前最大的分类序号时，将页面里选中的上级分类的编号也传给该后台服务，以便仅查询这个上级分类里的所有直系子类别（不包括子类别里面的子子类别）的最大序号，而不是查全部分类的最大序号，因为序号只是用来对某个类别里的直系子类别进行排序用的，不是同一个上级分类里的子类别互相之间的序号是没有任何关系的。

另外，在此将上级分类组件作为请求参数添加到表单的数据查询设置里后，当上级分类组件的值发生了变化后，就会自动重新查询对应分类里子分类的最大序号。



设置好后点确定按钮，完成设置。

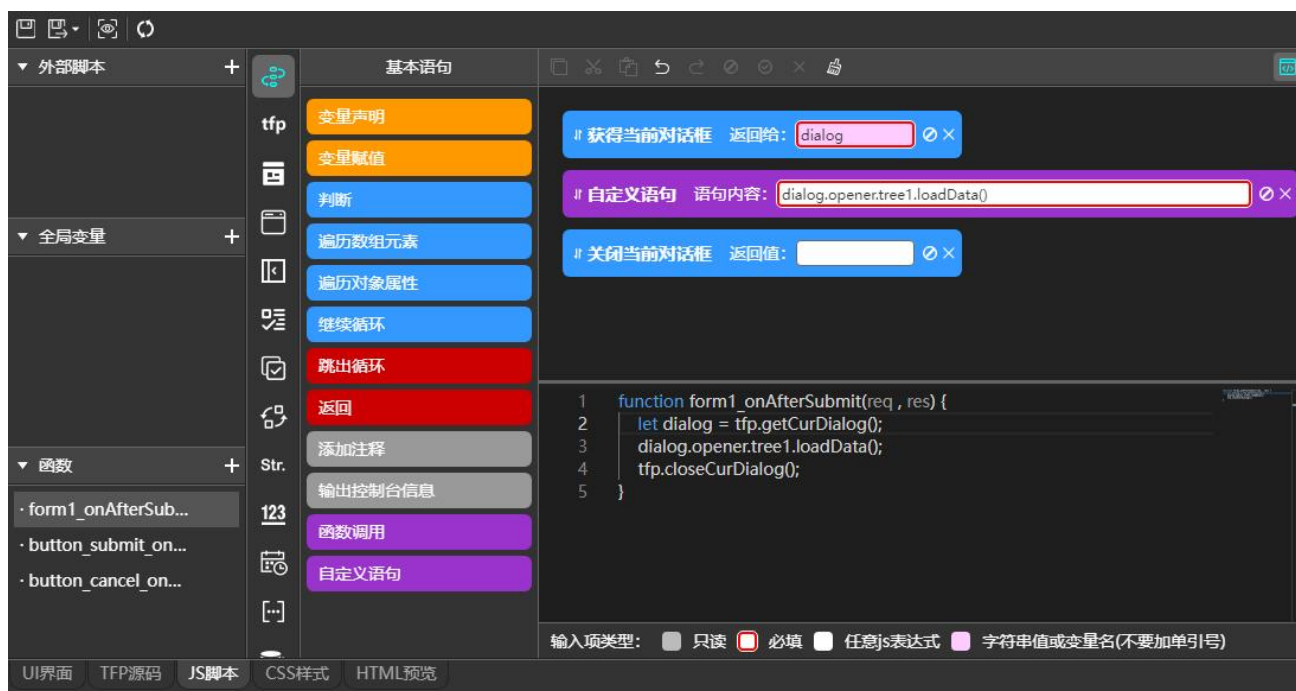
#### 4.6.5.3 设置分类序号输入项的数据绑定格式

然后选中分类序号输入项（组件编号为：sort\_num），在属性设置面板中，将“数据绑定格式”设置为：{sort\_num}，表示当打开该对话框时，根据表单的数据查询设置自动查询目前最大的分类序号（再加 1），并将其值绑定到分类序号这个输入项里。

#### 4.6.6 设置表单数据提交后事件

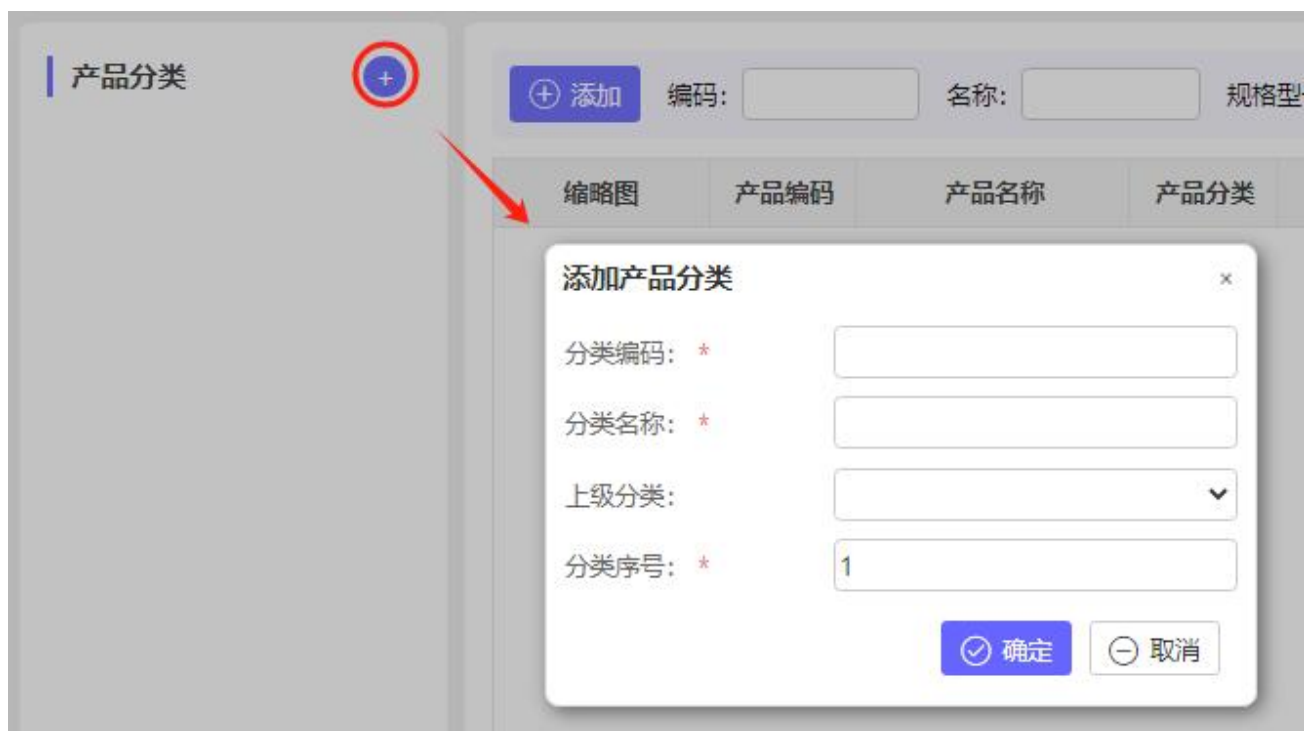
在添加产品分类的对话框页面中，选中表单组件 form1，然后双击事件设置面板中的“提交后”事件的输入框，打开该事件的可视化编排界面，在此，向导为我们自动创建了三条语句，其中第二条语句默认是让数据列表页面里的数据表格组件 grid1 重新加载数据，这里我们要改成如下内容，表示在添加完产品分类信息后，要重新加载列表页面里产品分类的树形列表：

```
dialog.opener.tree1.loadData()
```



#### 4.6.7 预览测试添加产品分类对话框

在 TaskBuilder 内打开产品信息列表页面，点击工具栏上的预览按钮，打开该页面的预览窗口，然后点击该页面左侧面板上部的添加产品分类的按钮，打开如下所示的对话框：



在此可以看到分类序号默认为 1，上级分类的下拉列表里也是空的，因为目前还没有添加

任何数据，我们输入分类编码和分类名称，例如 01 和日用品，然后点确定按钮，即可添加一个一级产品分类，如下图：



如果我们再打开这个添加产品分类的对话框，就可以看到分类序号变成了 2，而且上级分类的下拉列表里也有了数据，因为数据库里已经有一条分类数据了，如果我们选择这个分类作为上级分类，则下面的分类序号又变成了 1，因为该分类里还没有任何子分类。

#### 4.6.8 完善添加产品分类信息的后台服务

上面用向导创建的添加产品分类的后台服务在实际使用时是存在问题的，就是可以重复录入相同编码或名称的分类信息，所以需要对该服务进行完善，具体操作如下：

在销售管理示例项目的后台服务中，打开 product 文件夹里的 addType\_form1\_add.tbs，如果在该文件夹没有看到这个服务文件，可以在 product 文件夹上点右键，然后点刷新。

在该服务中，需要在添加分类信息前，先查询一下要添加的分类编码和名称是否已经存在，如果已经存在，则返回错误提示，不允许添加。

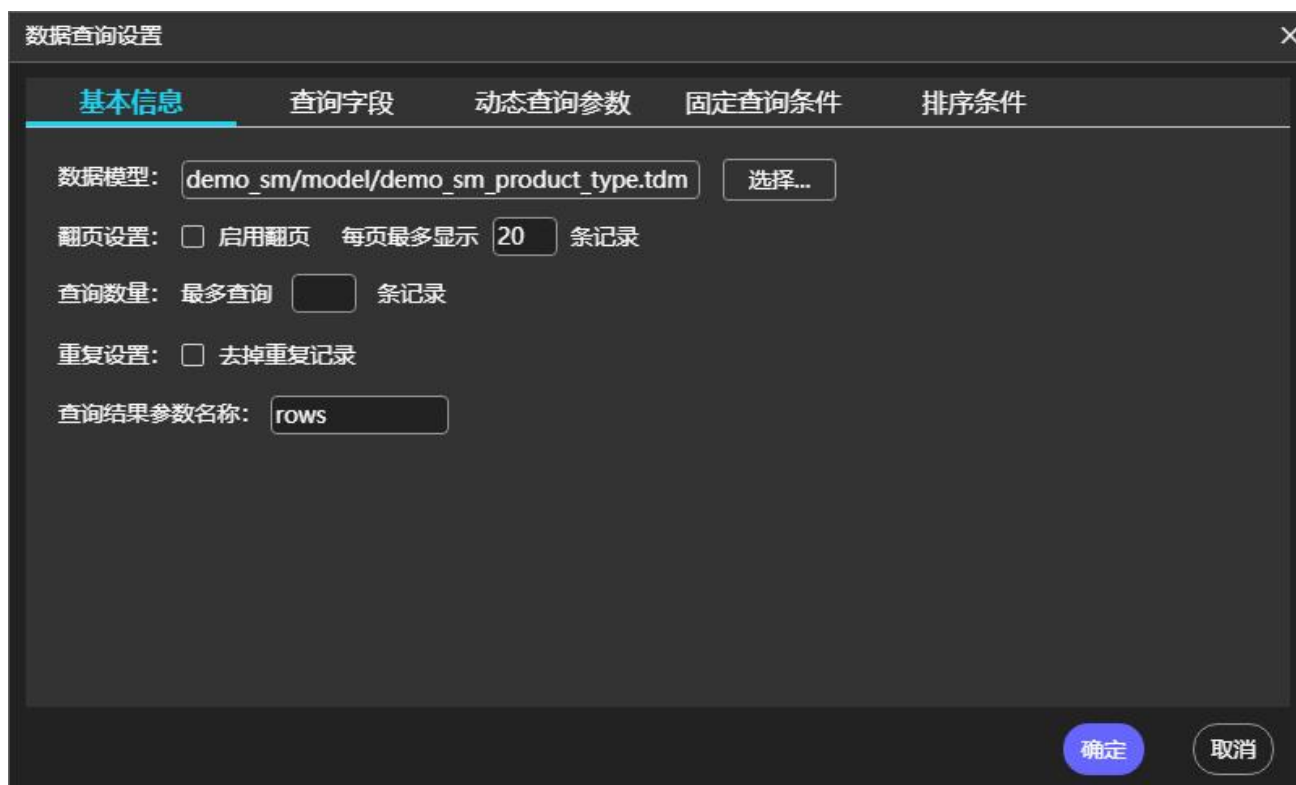
调整完的后台服务界面如下图所示：



上图的语句说明如下：

(1) 首先，从数据库操作中拖拽一个数据查询操作，点击设置，打开如下对话框：

在基本信息里，先选择产品分类的数据模型：demo\_sm\_product\_type.tdm



然后点击查询字段，任意选择一个字段，例如 id 字段，如下图：





然后点击固定查询条件，将条件设置为：

`(code={req.code} or (parent_id={req.parent_id} and name={req.name}))`

该条件表示查询分类编码等于添加页面里输入的编码，或者查询页面里选择的上级分类里分类名称等于页面里输入的分类名称的记录，如下图：



点确定保存设置。

(2) 然后再从基本语句里拖拽一条“判断”语句到上面的数据查询语句里，并将判断条件设置为：`rows.length>0`，表示如果查询到的记录数量大于 0，就说明已经存在重复记录，则需要返回错误提示。

(3) 从基本语句里拖拽一条“返回逻辑错误”语句到判断语句里，并将错误提示设置为：“分类编码和名称不能重复！”。

(4) 然后将原来的数据添加操作拖拽到判断语句下面，表示如果没有重复的记录，则正常执行数据添加操作。

这样设置后，在添加产品分类信息时，就不会出现录入重复数据的情况了。但还有一个问题，就是用向导创建产品分类信息的添加页面时，该页面里的表单组件的数据提交采用的基于“数据模型”的方式，这种方式在修改提交参数后，再保存页面时会自动重新生成对应的后台服务，就会造成我们上面所做的修改内容全部丢失，为了避免出现这种情况，需要修改表单组件的数据提交方式，操作方法如下：

选中表单组件 form1，点击属性设置里的“数据提交设置”，将顶部的“提交数据方式”改成“后台服务”，然后点击“选择...”按钮，选择前面这个我们修改过的添加产品分类信息的后台服务：`demo_sm/service/product/addType_form1_add.tbs`，然后点确定按钮，然后再保存页面，这时就不会再自动生成该后台服务，可以确保我们所做的个性化修改一直有效。

后续再遇到这种需要修改自动生成的后台服务的情况，都应该按照这种方式处理，包括数据查询、提交、删除、导入、导出等。

数据提交设置

提交数据方式：☐ 数据模型 ☐ 数据表单 ☒ 后台服务 ☐ SQL语句 ☐ HTTP请求

后台服务

请求参数

☒ 直接从后台选择指定的后台服务

demo\_sm/service/product/addType\_form1\_add.i

选择...

☐ 使用页面上已有的服务请求组件

清空

确定

取消

我们再预览测试，添加编码或名称重复的产品分类，就会收到错误提示，如下图：



至此，我们就完成了添加产品分类的功能。

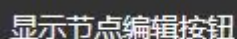
## 4.7 实现修改产品分类的功能

接下来我们实现修改产品分类的功能。

## 4.7.1 设置树形组件节点编辑按钮

### 4.7.1.1 启用显示节点编辑按钮功能

为了实现产品分类信息的修改，需要在树形组件的属性里，将“显示节点编辑按钮”选项打开，表示当鼠标放到节点上时，右侧就会出现编辑按钮，点击该按钮就可以调用对应的事件处理函数，打开对话框修改产品分类信息。



### 4.7.1.2 设置节点编辑按钮点击事件

选中树形组件，在右侧的事件设置面板里，双击“点击节点的编辑按钮时”事件输入框，打开事件脚本编辑器，从左侧语句库的对话框操作中拖拽“打开对话框”语句到右边编辑区，然后点击该语句上的“参数设置”按钮，打开下图所示对话框：

参数设置

×

语句类型：

打开对话框

| 参数名    | 必填 | 参数类型 | 参数值                     |
|--------|----|------|-------------------------|
| 标题     | 是  | 表达式  | '修改产品分类'                |
| 路径     | 是  | 表达式  | 'updateType.tfp?id='+id |
| 宽度     | 是  | 表达式  | '400px'                 |
| 高度     | 是  | 表达式  | '260px'                 |
| 左边距    | 否  | 表达式  |                         |
| 上边距    | 否  | 表达式  |                         |
| 隐藏关闭按钮 | 否  | 布尔值  | ▼                       |
| 禁止移动   | 否  | 布尔值  | ▼                       |
| 传入参数   | 否  | 变量名称 |                         |
| 回调参数名  | 否  | 字符串值 | ret                     |

如果参数类型是表达式，则可输入任意js表达式，这时如果输入字符串，两边需要加单引号；如果参数类型是字符串值，则直接输入字符串即可，两边不需要加单引号。

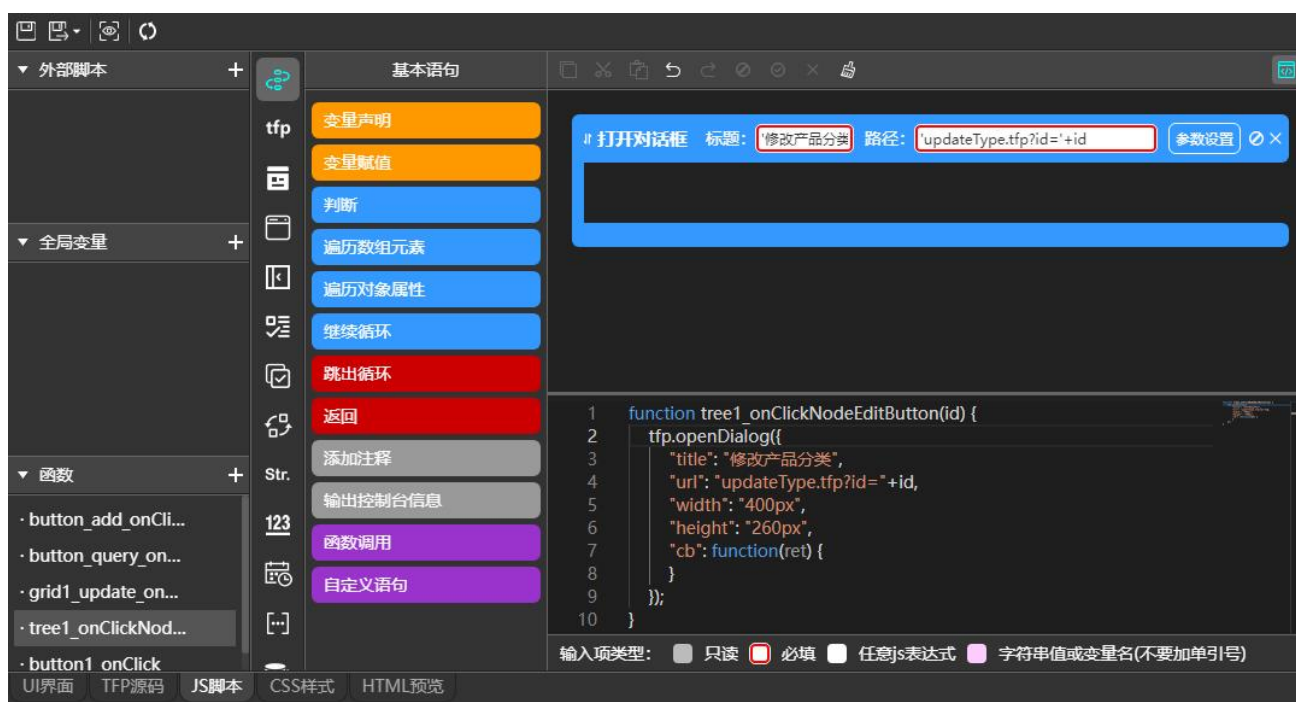
确定

取消

在里面设置好以下参数，然后点确定：

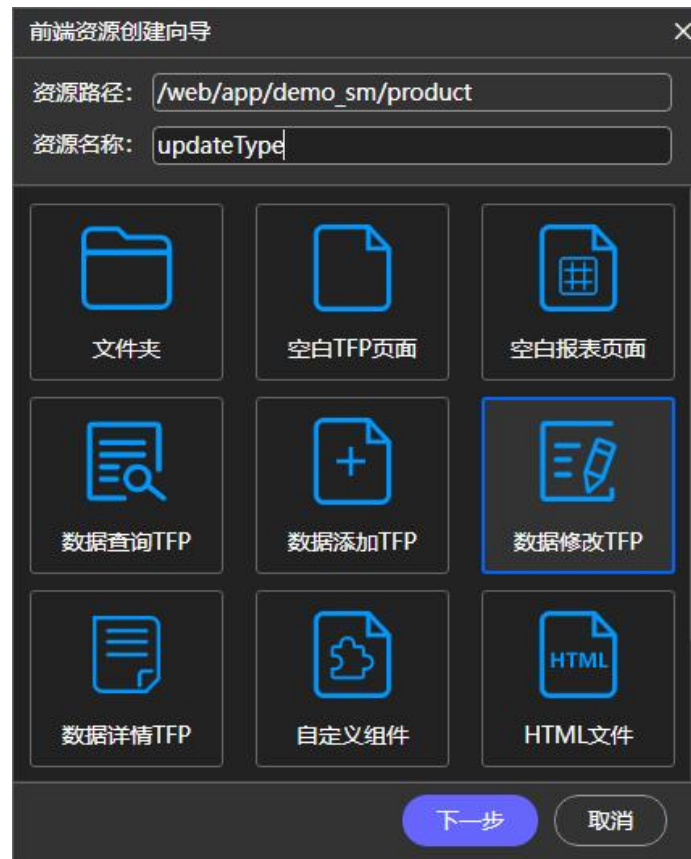
- **标题：** '修改产品分类'
- **路径：** 'updateType.tfp?id='+id
- **宽度：** '400px'
- **高度：** '260px'

上图对话框中的参数值输入框里可以设置任意 JavaScript 表达式、变量或具体的值，如果是字符串值，需要在两边加英文的单引号。其中，路径中的 updateType.tfp 是我们稍后要创建的产品分类修改页面，id 参数是在点击某个产品分类节点的编辑按钮时，传给事件函数的当前产品分类的 id 值。

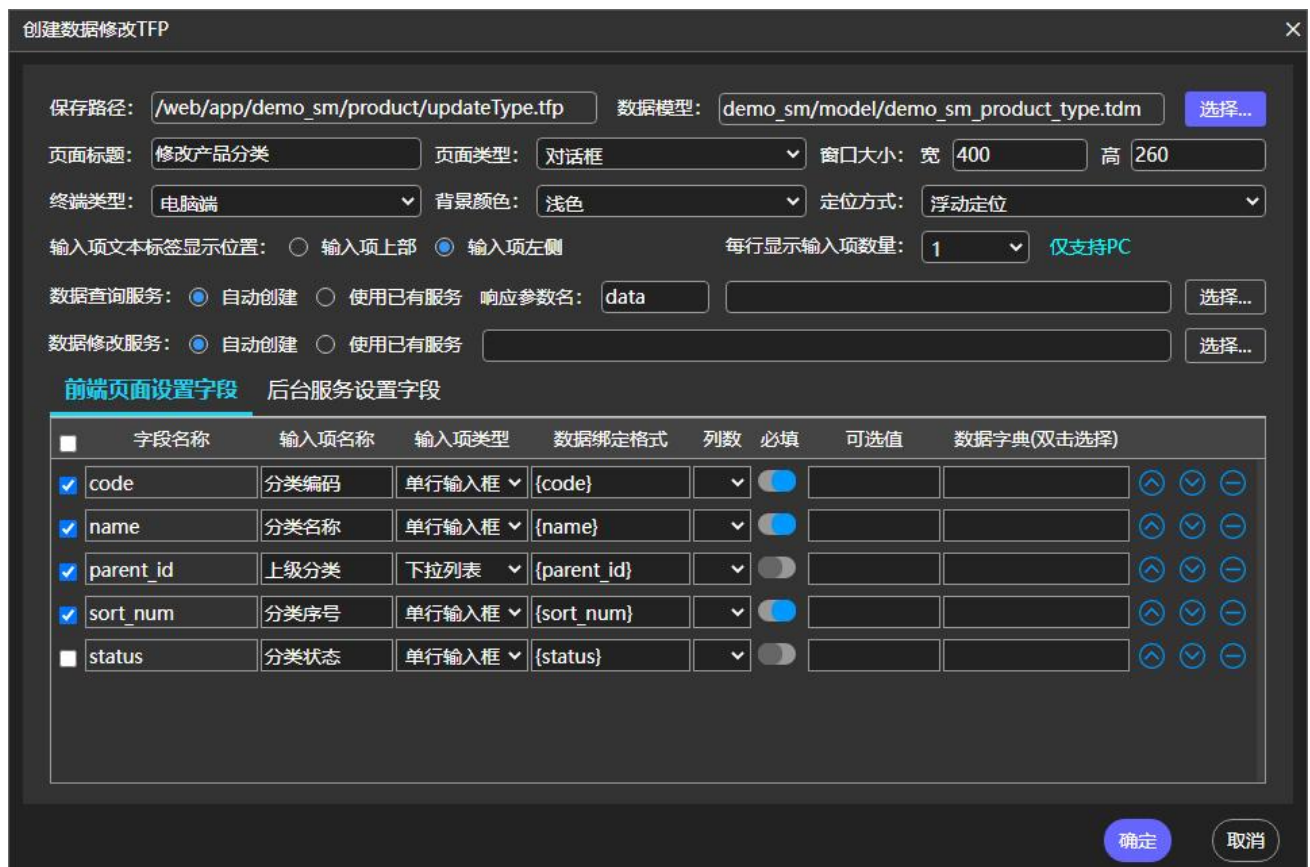


## 4.7.2 创建修改产品分类的对话框

在销售管理示例项目的前端文件里，点击 product 目录右侧的加号按钮，打开下图所示的“前端资源创建向导”，在里面选中“数据修改 TFP”，资源名称输入：updateType：



然后点下一步，打开下图所示的“创建数据修改 TFP”对话框：





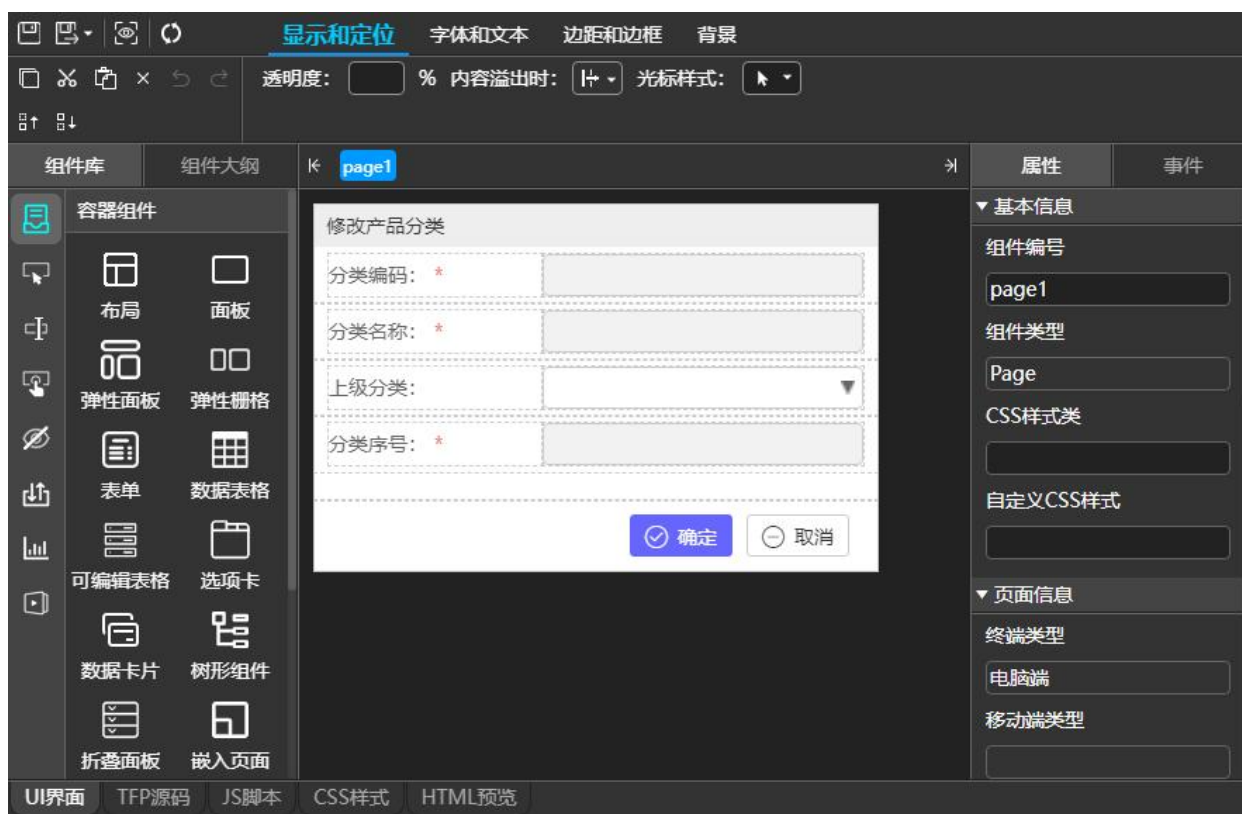
在该对话框中，先设置好基本信息，需要特殊设置的信息包括：

- **数据模型**：选择产品分类信息数据模型：emo\_sm\_product\_type.tdm
- **页面标题**：设置为修改产品分类；
- **页面类型**：设置为对话框；
- **窗口大小**：宽度设置为 400，高度设置为 260（会根据其他选项会自动设置）；
- **输入项文本标签显示位置**：输入项左侧
- **每行显示输入项数量**：设置为 1

然后在下面的“前端页面设置字段”中，选择所有字段，并按下表设置各个字段：

| 字段名称      | 输入项名称 | 输入项类型 | 必填 | 可选值 |
|-----------|-------|-------|----|-----|
| code      | 分类编码  | 单行输入框 | 是  |     |
| name      | 分类名称  | 单行输入框 | 是  |     |
| parent_id | 上级分类  | 下拉列表  |    |     |
| sort_num  | 分类序号  | 单行输入框 | 是  |     |

然后点确定，完成修改页面的创建，界面如下所示：



### 4.7.3 设置上级分类下拉列表的数据查询参数

在上面创建的修改产品分类的对话框页面里，选中上级分类下拉列表组件（编号为：parent\_id），然后点击属性里的“数据查询设置”，打开如下图所示的对话框：



数据查询设置对话框，包含以下配置项：

- 查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求
- 数据模型：请求参数 查询条件 分组和排序
- 最多查询： 条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入
- 数据模型路径：
- 
- 字段列表：

| <input type="checkbox"/>            | 字段名称      | 字段备注   | 字段格式(SQL, 双击可编辑) | 字段别名 | <input type="button" value="⊖"/> |
|-------------------------------------|-----------|--------|------------------|------|----------------------------------|
| <input checked="" type="checkbox"/> | id        | 分类编号   |                  |      | <input type="button" value="⊖"/> |
| <input type="checkbox"/>            | code      | 分类编码   |                  |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | name      | 分类名称   |                  |      | <input type="button" value="⊖"/> |
| <input checked="" type="checkbox"/> | parent_id | 上级分类编号 |                  |      | <input type="button" value="⊖"/> |
| <input type="checkbox"/>            | sort_num  | 分类序号   |                  |      | <input type="button" value="⊖"/> |
| <input type="checkbox"/>            | status    | 分类状态   |                  |      | <input type="button" value="⊖"/> |

在此，先选择产品分类数据模型：demo\_sm\_product\_type.tdm，然后在下面的字段列表里选择 id、name 和 parent\_id 字段，然后再点击“请求参数”选项卡，在里面添加一个名为 id 的参数，参数类型设置为“URL 参数”，表示该参数的值是从页面的 URL 里获取的，如下图：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '请求参数' (Request Parameters) tab selected. The '查询数据方式' (Query Data Method) is set to '数据模型' (Data Model). The '参数名' (Parameter Name) is 'id', the '参数类型' (Parameter Type) is 'URL 参数' (URL Parameter), and the '参数值' (Parameter Value) is empty. There are buttons for adding (+), up, down, and delete (-) parameters. A green arrow points to the '参数值' field with the text '↓ 在此设置需要传递给后台的查询参数' (↓ Set the query parameters to be passed to the backend here). At the bottom are '清空' (Clear), '确定' (Confirm), and '取消' (Cancel) buttons.

| 参数名 | 参数类型   | 参数值 |
|-----|--------|-----|
| id  | URL 参数 |     |

然后再次点击“查询条件”选项卡，在里面输入查询条件：`id<>{req.id} and status=0`，表示不查询当前要修改的产品分类，以免将当前分类本身作为自己的上级分类，从而出现逻辑错误：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '查询条件' (Query Conditions) tab selected. The '查询数据方式' (Query Data Method) is still '数据模型' (Data Model). The '固定查询条件' (Fixed Query Conditions) section shows '查询字段' (Query Field) as 'id' and '查询条件' (Query Condition) as '<>{req.id} and status=0'. The '动态查询条件' (Dynamic Query Conditions) section is empty. At the bottom are '清空' (Clear), '确定' (Confirm), and '取消' (Cancel) buttons.

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

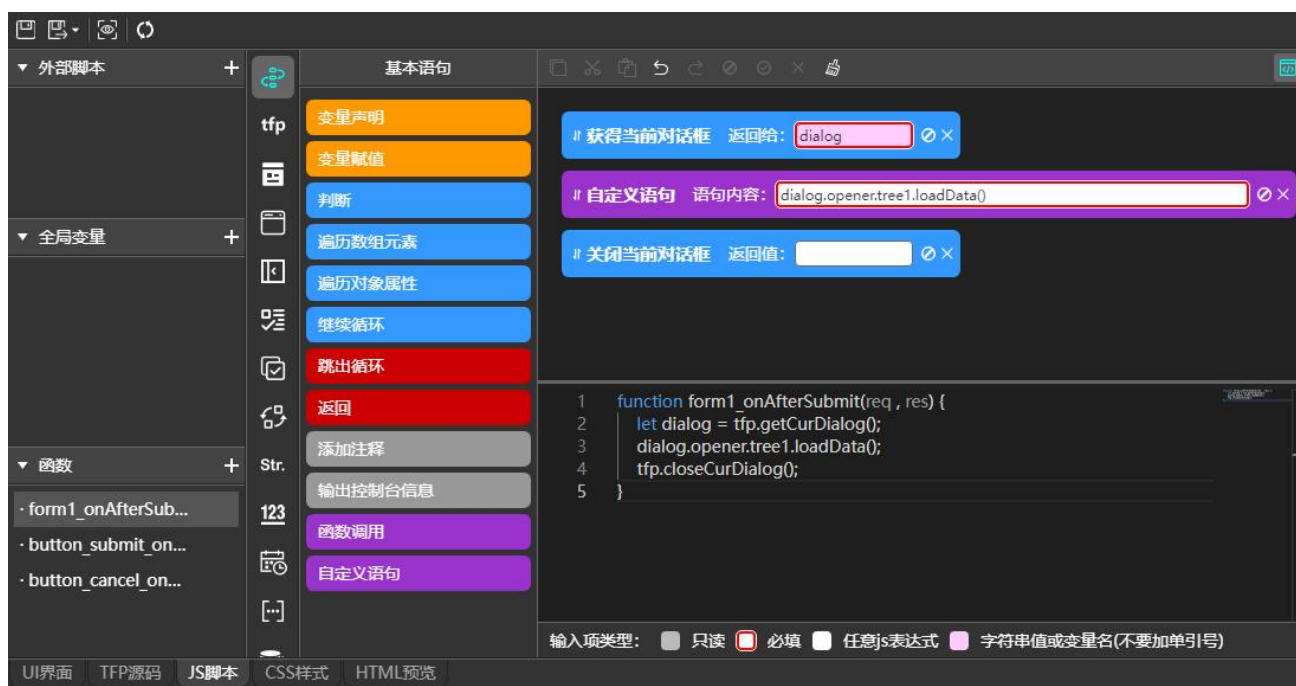
都设置好后，点确定按钮，然后在该组件的属性里，继续设置以下属性：

- **选项值字段**：设置为 id；
- **选项文本字段**：设置为 name；
- **显示树形结构**：开启该选项；
- **父节点主键字段**：设置为 parent\_id；
- **根节点主键值**：设置为 0；

以上信息都设置好后，打开修改产品分类信息对话框时，就会自动查询已有的产品分类。

#### 4.7.4 设置表单数据提交后事件

在上面创建的修改产品分类的对话框页面里，选中表单组件 form1，然后双击事件设置面板里的“提交后”事件输入框，进入该事件的可视化编排界面，该界面中的语句都是向导自动生成的，里面第二条语句默认是重新加载列表页面里数据表格的数据，需要改成重新加载页面左侧产品分类树形列表里的数据，语句内容需要改成：`dialog.opener.tree1.loadData()`，最终效果如下图所示：



## 4.7.5 完善修改产品分类的后台服务

与添加产品分类的后台服务一样，用向导创建的修改产品分类的后台服务存在一个问题，就是没有对录入的数据进行唯一性校验，用户可能会录入重复的数据，为此需要再完善一下，具体操作如下：

在销售管理项目的后台服务中，打开 product 文件夹里的 updateType\_form1\_update.tbs，如果在该文件夹内没有看到这个服务文件，可以在 product 文件夹上点右键，然后点刷新。

在该服务中，需要在修改分类信息前，先查询一下要修改的分类编码和名称是否已经存在，如果已经存在，则返回错误提示，不允许添加。

调整完的后台服务界面如下图所示：



上图的语句说明如下：

(1) 首先，从基本语句里拖拽一条变量声明语句，变量名设置为：parentId，初始值设置为：0，表示默认没有设置上级分类；

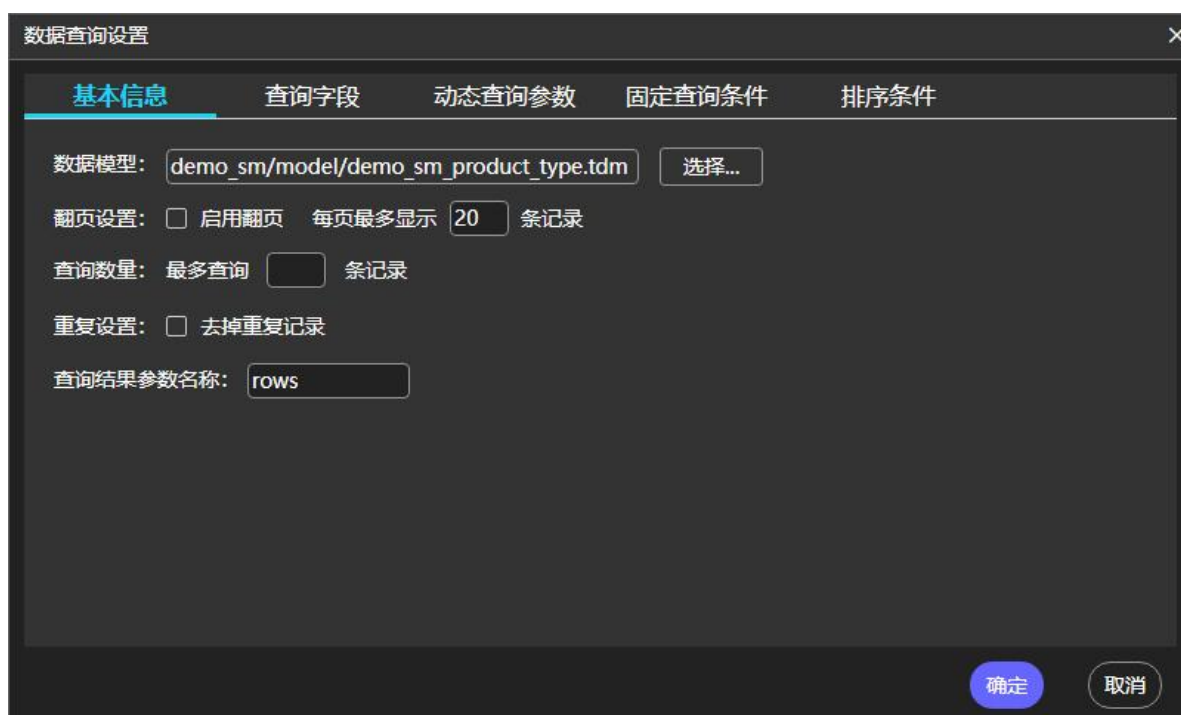
(2) 然后从基本语句里拖拽一个判断语句，将判断条件设置为：req.parent\_id，表示判断前端是否发送了请求参数：parent\_id；

(3) 然后再从基本语句里拖拽一条变量赋值语句到上面的判断语句中，将前端请求参数 req.parent\_id 赋值给变量 parentId；

通过上面这三条语句，可以实现如果前端没有传上级分类编号时，变量 parentId 的值为 0，如果前端传了上级分类编号，则将变量 parentId 的值设置为前端传的上级分类编号，这样后续在查询和修改分类信息时，就可以保证上级分类编号始终都是有值的，而不是空的。

(4) 从数据库操作中拖拽一个数据查询操作，点击设置，打开如下对话框：

在基本信息里，先选择产品分类的数据模型：demo\_sm\_product\_type.tdm



然后点击“查询字段”选项卡，在这里随意选择一个字段，例如 id 字段，如下图：

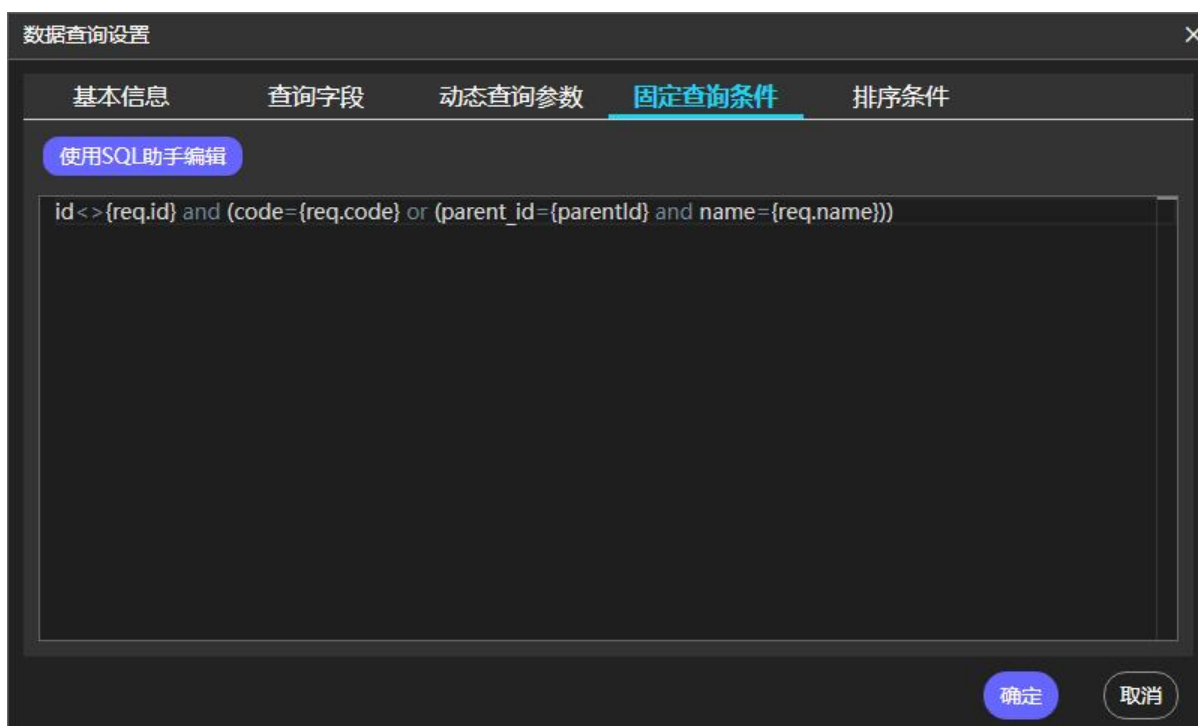




然后点击“固定查询条件”，在此输入以下查询条件：

```
id<>{req.id} and (code={req.code} or (parent_id={parentId} and name={req.name}))
```

表示查询 id 不等于当前要修改的产品分类的 id，并且，分类编码等于修改页面里输入的编码，或者上级分类编号等于页面里选择的上级分类，且分类名称等于页面里输入的分类名称的记录，这句话有点绕，但总体意思就是修改时，编码和名称不能重复。



都设置好后，点确定保存。

(5) 然后再从基本语句里拖拽一条“判断”语句到上面的数据查询语句里，并将判断条件设置为：`rows.length>0`，表示如果查询到的记录数量大于0，就说明已经存在重复记录，则需要返回错误提示。

(6) 从基本语句里拖拽一条“返回逻辑错误”语句到判断语句里，并将错误提示设置为：“产品分类编码和名称不能重复！”。

(7) 然后将原来的数据修改操作拖拽到判断语句下面，表示如果没有重复的记录，则正常执行数据修改操作，但是还语句需要修改一下，点击设置，打开设置对话框，将 `parent_id` 字段的值修改为：`parentId`，表示从这个变量里取值，而不是从 `req.parent_id` 里取值，因为前端页面里，可能没有选择上级分类，`req.parent_id` 的值可能是空的。



然后点确定，然后保存整个后台服务。

## 4.7.6 修改表单的数据提交设置

在后续修改表单的数据提交设置时，可能会自动重新生成修改产品分类的后台服务，那我们上面对该服务所做的修改就有可能会被覆盖，为了避免出现这种情况，需要修改表单的数据提交方式，具体操作如下：

选中表单组件 form1，点击属性设置里的“数据提交设置”，将顶部的“提交数据方式”改成“后台服务”，然后点击“选择...”按钮，选择前面这个我们修改过的修改产品分类信息的后台服务：demo\_sm/service/product/updateType\_form1\_update.tbs，然后点确定按钮，然后再保存页面，这时就不会再自动生成该后台服务，可以确保我们所做的个性化修改一直有效。

数据提交设置

提交数据方式：☐ 数据模型 ☐ 数据表单 ☒ 后台服务 ☐ SQL语句 ☐ HTTP请求

**后台服务** 请求参数 修改条件

☒ 直接从后台选择指定的后台服务

☐ 使用页面上已有的服务请求组件

### 4.7.7 预览测试修改产品分类对话框

在 TaskBuilder 内打开产品信息列表页面，点击工具栏上的预览按钮，打开该页面的预览窗口，然后把鼠标放在该页面左侧面板上产品分类树形列表的某个分类名称上，就会在该分类右侧显示编辑按钮，如下图所示：



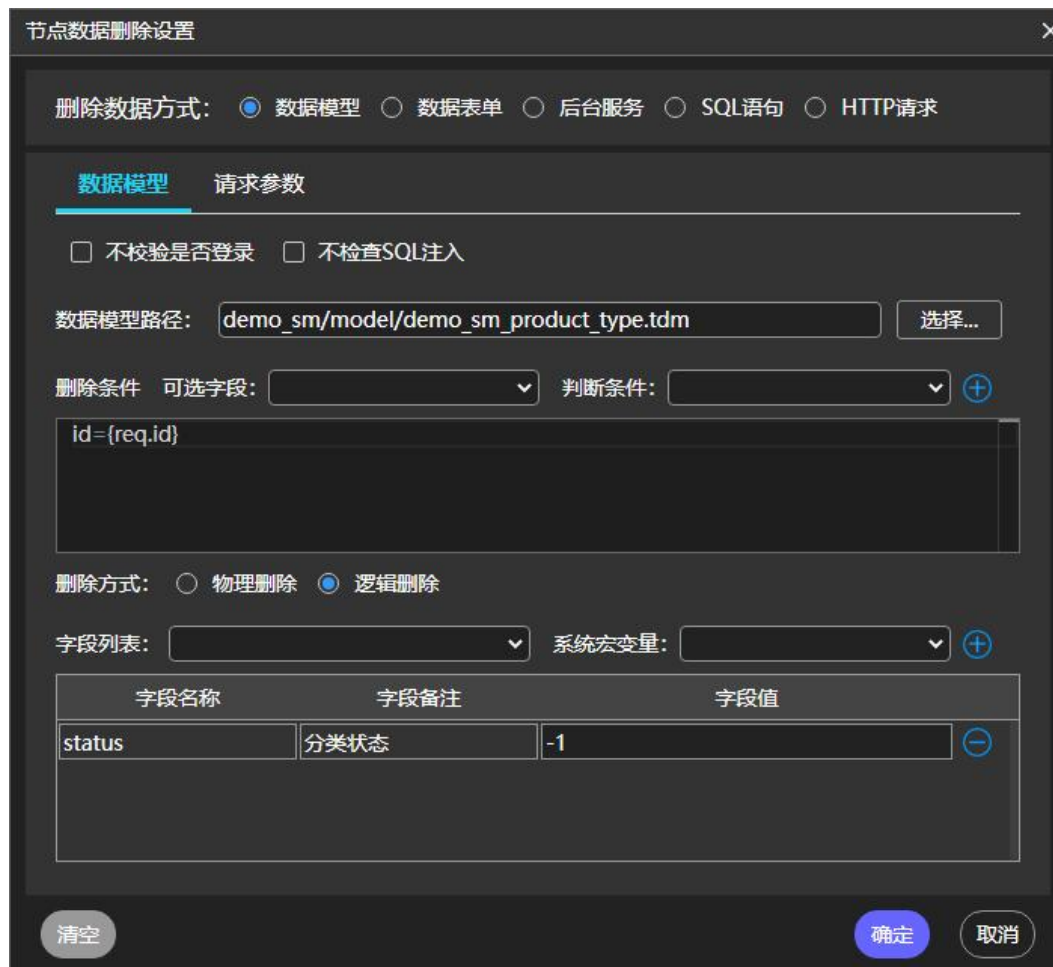
点击该编辑按钮，会打开如下所示的修改产品分类的对话框：

在此可以修改产品分类信息，然后点确定按钮，就可以发现列表页面里的产品分类发生了相应的变化。

## 4.8 实现删除产品分类的功能

### 4.8.1 设置树形组件数据删除参数

在产品信息列表页面里，点击树形组件属性里的“节点数据删除设置”，打开如下对话框：



节点数据删除设置

删除数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据模型** 请求参数

☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径: demo\_sm/model/demo\_sm\_product\_type.tdm 选择...

删除条件 可选字段: 判断条件: +

id={req.id}

删除方式: ☐ 物理删除 ☒ 逻辑删除

字段列表: 系统宏变量: +

| 字段名称   | 字段备注 | 字段值  |
|--------|------|------|
| status | 分类状态 | -1 - |

清空 确定 取消

删除条件设置为: `id={req.id}`, 删除方式设置为: 逻辑删除, 并将 `status` 字段添加到下面的字段列表里, 把字段值设置为: -1, 表示删除时, 将该字段的值修改为-1, 实现逻辑删除。

## 4.8.2 启用显示节点删除按钮功能

为了实现产品分类信息的删除, 需要在树形组件的属性里, 将“显示节点删除按钮”开关选项打开, 表示当鼠标放到节点上时, 右侧就会出现删除按钮, 点击该按钮就可以调用上面设置的删除服务删除该节点的数据。



## 4.8.3 预览测试删除产品分类功能

在 TaskBuilder 内打开产品信息列表页面, 点击工具栏上的预览按钮, 打开该页面的预览

窗口，然后把鼠标放在该页面左侧面板上产品分类树形列表的某个分类名称上，就会在该分类右侧显示删除按钮，如下图所示：



点击该按钮，会弹出是否确定删除的提示按钮：



点击确定，即可实现产品分类信息的删除。

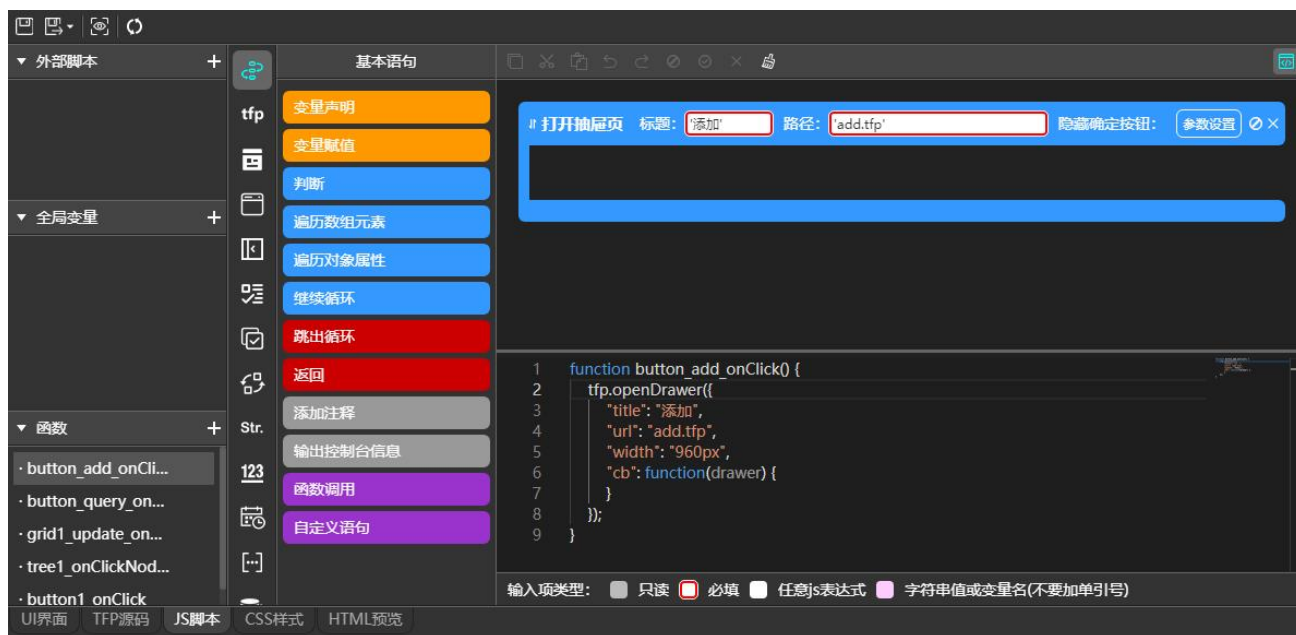
## 4.9 实现添加产品信息的功能

接下来，我们实现添加产品的功能。

### 4.9.1 设置打开添加产品信息抽屉页的按钮

在产品信息列表页面里，选中产品列表上面的“添加”按钮，然后双击事件设置面板里的“点击时”事件输入框，打开该事件的可视化编排界面，可以看到，向导已经为我们创建好了打开添加产品信息抽屉页的相关语句，界面如下图所示：

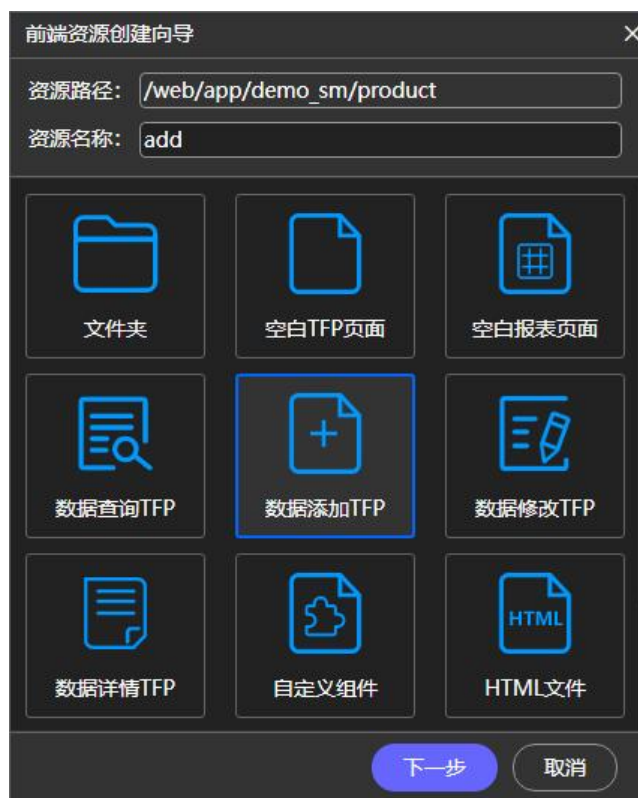




在这里，我们不需要做任何修改，使用向导默认创建的语句即可。

## 4.9.2 创建添加产品信息抽屉页

在销售管理示例项目的前端文件里点击 product 目录右侧的加号按钮,打开下图所示的“前端资源创建向导”，在里面选中“数据添加 TFP”，资源名称输入：add：



然后点击下一步按钮，打开如下图所示的“创建数据添加 TFP”对话框：

创建数据添加TFP

保存路径：

/web/app/demo\_sm/product/add.tfp

数据模型：

demo\_sm/model/demo\_sm\_product.tdm

选择...

页面标题：

添加产品信息

页面类型：

抽屉页

窗口大小：

宽

高

终端类型：

电脑端

背景颜色：

浅色

定位方式：

浮动定位

输入项文本标签显示位置：

☒ 输入项上部

☐ 输入项左侧

每行显示输入项数量：

3

仅支持PC

数据添加服务：

☒ 自动创建

☐ 使用已有服务

选择...

前端页面设置字段

后台服务设置字段

| <input checked="" type="checkbox"/> | 字段        | 输入项名称 | 输入项类型 | 列数 | 必填                                  | 可选值 | 数据字典(双击选择) | 默认值  |
|-------------------------------------|-----------|-------|-------|----|-------------------------------------|-----|------------|------|
| <input checked="" type="checkbox"/> | code      | 产品编码  | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            |      |
| <input checked="" type="checkbox"/> | name      | 产品名称  | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            |      |
| <input checked="" type="checkbox"/> | type_id   | 产品分类  | 下拉列表  |    | <input checked="" type="checkbox"/> |     |            |      |
| <input checked="" type="checkbox"/> | model     | 规格型号  | 单行输入框 |    | <input type="checkbox"/>            |     |            |      |
| <input checked="" type="checkbox"/> | unit      | 计量单位  | 单行输入框 |    | <input checked="" type="checkbox"/> |     | unit       |      |
| <input checked="" type="checkbox"/> | price     | 单价    | 单行输入框 |    | <input type="checkbox"/>            |     |            | 0.00 |
| <input checked="" type="checkbox"/> | thumbnail | 缩略图   | 图片上传  | 3  | <input checked="" type="checkbox"/> |     |            |      |
| <input checked="" type="checkbox"/> | comment   | 备注    | 多行输入框 | 3  | <input type="checkbox"/>            |     |            |      |

确定

取消

在此，先设置好数据模型，选择前面创建的产品信息数据模型：demo\_sm\_product.tdm，页面标题设置为：添加产品信息，其他参数不用动，用默认的即可。

然后在下面的“前端页面设置字段”里，选中下表中的字段，并设置相关参数：

| 字段名       | 输入项名称 | 输入项类型 | 列数 | 必填 | 数据字典 | 默认值  |
|-----------|-------|-------|----|----|------|------|
| code      | 产品编码  | 单行输入项 |    | 是  |      |      |
| name      | 产品名称  | 单行输入项 |    | 是  |      |      |
| type_id   | 产品分类  | 下拉列表  |    | 是  |      |      |
| model     | 规格型号  | 单行输入项 |    |    |      |      |
| unit      | 计量单位  | 单行输入项 |    | 是  | unit |      |
| price     | 单价    | 单行输入项 |    |    |      | 0.00 |
| thumbnail | 缩略图   | 图片上传  | 3  |    |      |      |
| comment   | 备注    | 多行输入框 | 3  |    |      |      |

设置好上面这些字段信息后，点击“后台服务设置字段”，在此添加字段 status，并将其

值设置为 0，表示新添加的产品信息默认是有效的。

创建数据添加TFP

保存路径: /web/app/demo\_sm/product/add.tfp 数据模型: demo\_sm/model/demo\_sm\_product.tdm 选择...

页面标题: 添加产品信息 页面类型: 抽屉页 窗口大小: 宽 高

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧 每行显示输入项数量: 3 仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务 选择...

前端页面设置字段 后台服务设置字段

字段列表: 系统宏变量: +

| 字段名    | 字段值(JS表达式) |
|--------|------------|
| status | 0 -        |

确定 取消

都设置好后点击确定按钮，完成添加产品信息抽屉页的创建，界面如下图所示：

显示和定位 字体和文本 边距和边框 背景

透明度: % 内容溢出时: 光标样式:

组件库 组件大纲 K page1 属性 事件

容器组件

- 布局
- 弹性面板
- 弹性栅格
- 表单
- 数据表格
- 可编辑表格
- 选项卡
- 数据卡片
- 树形组件
- 折叠面板
- 嵌入页面
- 宫格
- 宫格行

产品编码: \* 产品名称: \* 产品分类: \*

规格型号: 计量单位: \* 单价: 0

缩略图: 上传图片

备注:

▼ 基本信息

组件编号 page1

组件类型 Page

CSS样式类

自定义CSS样式

▼ 页面信息

终端类型 电脑端

移动端类型

组件默认定位方式 浮动定位

UI界面 TFP源码 JS脚本 CSS样式 HTML预览

### 4.9.3 设置产品分类下拉列表的数据查询参数

在添加产品信息页面里，选中产品分类的“下拉列表”组件，然后点击属性里的“数据查询设置”，打开如下所示对话框：



数据查询设置对话框，包含以下配置项：

- 查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求
- 数据模型：请求参数 查询条件 分组和排序
- 最多查询  条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入
- 数据模型路径：
- 
- 字段列表：

| <input type="checkbox"/>            | 字段名称      | 字段备注   | 字段格式(SQL, 双击可编辑) | 字段别名 |   |
|-------------------------------------|-----------|--------|------------------|------|---|
| <input checked="" type="checkbox"/> | id        | 分类编号   |                  |      | ⊖ |
| <input type="checkbox"/>            | code      | 分类编码   |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | name      | 分类名称   |                  |      | ⊖ |
| <input checked="" type="checkbox"/> | parent_id | 上级分类编号 |                  |      | ⊖ |
| <input type="checkbox"/>            | sort_num  | 分类序号   |                  |      | ⊖ |
| <input type="checkbox"/>            | status    | 分类状态   |                  |      | ⊖ |

底部按钮：

在该对话框中，先选择数据模型：demo\_sm\_product\_type.tdm，然后在下面的字段列表中选中 id、name 和 parent\_id 这三个字段，然后点击上面的“查询条件”选项卡，在里面设置查询条件为：status=0，表示只查询有效的产品分类，如下图所示：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '查询条件' (Query Conditions) tab selected. The '查询数据方式' (Query Data Method) is set to '数据模型' (Data Model). Under '固定查询条件' (Fixed Query Conditions), the '查询字段' (Query Field) is empty and the '查询条件' (Query Condition) is 'status=0'. Under '动态查询条件' (Dynamic Query Conditions), the '查询字段' (Query Field) is empty and the '查询条件' (Query Condition) is empty. Below these is a table for '查询表达式(SQL, 双击可编辑)' (Query Expression (SQL, double-click to edit)).

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

Buttons at the bottom: 清空 (Clear), 确定 (Confirm), 取消 (Cancel).

然后再点击上面的“分组和排序”，在里面添加两个排序条件，即按 parent\_id（上级分类编号）和 sort\_num（分类序号）正序排列，如下图所示：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '分组和排序' (Grouping and Sorting) tab selected. The '查询数据方式' (Query Data Method) is set to '数据模型' (Data Model). Under '分组条件' (Grouping Conditions), the '分组字段' (Grouping Field) is empty. Under '排序条件' (Sorting Conditions), the '字段' (Field) is empty and the '排序规则' (Sorting Rule) is '正序' (Ascending). Below these is a table for '排序字段' (Sorting Field) and '排序规则' (Sorting Rule).

| 排序字段      | 排序规则 |
|-----------|------|
| parent_id | asc  |
| sort_num  | asc  |

Buttons at the bottom: 清空 (Clear), 确定 (Confirm), 取消 (Cancel).

然后点确定按钮，即可完成产品分类下拉列表组件的数据查询设置。

然后在该组件的属性里，继续设置以下属性：

- **选项值字段**：设置为 id；
- **选项文本字段**：设置为 name；
- **显示树形结构**：开启该选项；
- **父节点主键字段**：设置为 parent\_id；
- **根节点主键值**：设置为 0；

以上信息都设置好后，打开添加产品信息抽屉页时，就会自动查询产品分类信息，并显示在该列表内。

#### 4.9.4 设置缩略图上传组件有关参数

添加产品信息时，可以上传产品的缩略图，但向导生成的图片上传组件有些细节不够完善，需要继续设置。

在添加产品信息页面里，选中图片上传组件，在属性面板里设置以下属性：

- **应用代码**：应用代码用来区分各个应用上传的附件具体属于哪个应用，可以使用字母、数字或下划线，这里设置为 product\_thumbnail；
- **保持唯一**：将该选项打开，表示每个产品只需要保留一张缩略图，如果后续再上传修改，原有的图片附件信息就不再与该产品关联；
- **文件总数**：用来设置可以上传的文件总数，这里设置为 1，表示仅能上传一张缩略图；
- **图片显示宽度**：设置为 96px；
- **图片显示高度**：设置为 96px；
- **允许删除**：将该选项打开，表示可以删除已上传的图片，然后重新上传。



## 4.9.5 完善添加产品信息的后台服务

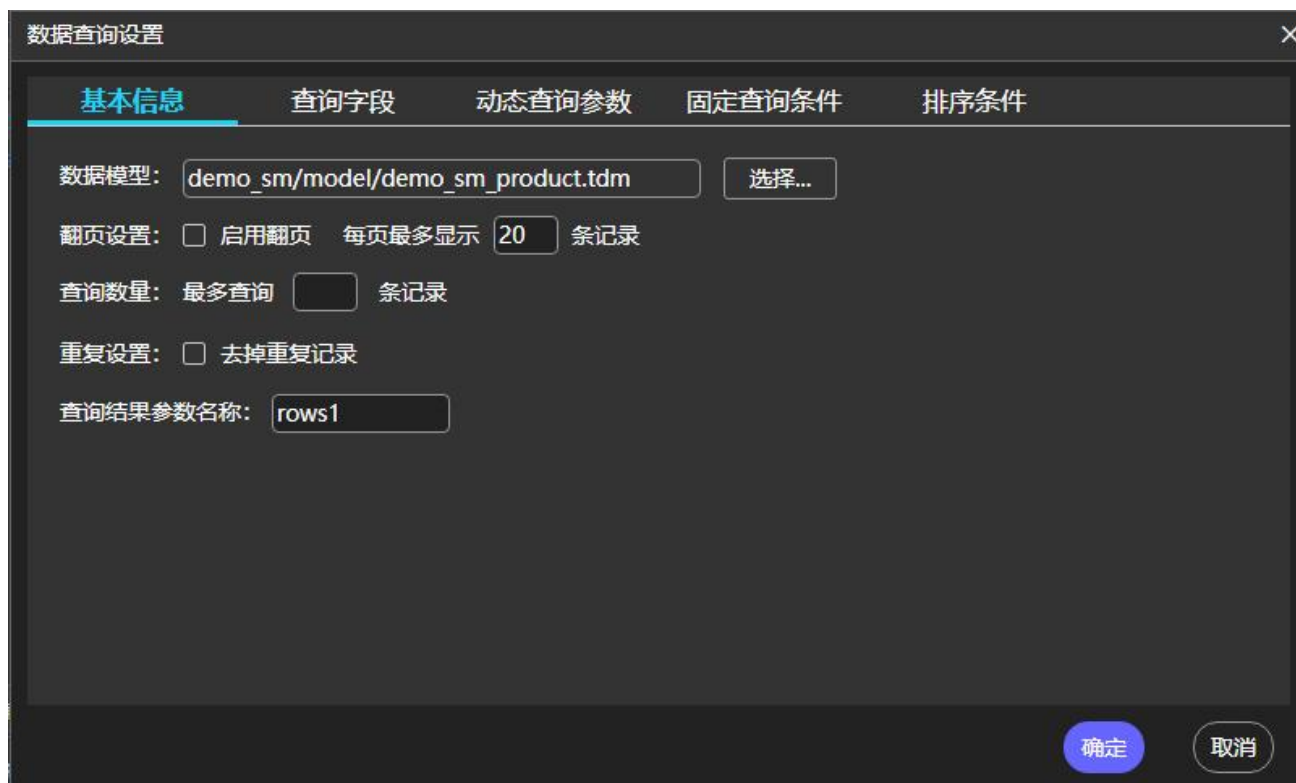
用向导自动创建的添加产品信息的后台服务存在以下问题，需要继续完善：

- (1) 产品编码和名称没有校验是否已存在，可能会录入重复的数据；
- (2) 产品序号没有赋值，需要自动生成；
- (3) 缩略图字段的值需要特殊设置；

在 TaskBuilder 内打开销售管理示例项目后台服务里 product 目录里的 add\_form1\_add.tbs，然后进行以下操作：

### 4.9.5.1 判断产品编码或名称是否存在

在语句库的数据库操作中，拖拽一个数据查询语句到最上面，然后点击该语句的设置按钮，先在基本信息里选择数据模型 demo\_sm\_product.tdm，然后设置查询结果参数名称：rows1：



数据查询设置

基本信息 查询字段 动态查询参数 固定查询条件 排序条件

数据模型: demo\_sm/model/demo\_sm\_product.tdm 选择...

翻页设置: ☐ 启用翻页 每页最多显示 20 条记录

查询数量: 最多查询 1 条记录

重复设置: ☐ 去掉重复记录

查询结果参数名称: rows1

确定 取消

然后点击“查询字段”选项卡，在里面选择字段 id：



然后点击“固定查询条件”选项卡，在里面设置好查询条件：

`(code={req.code} or name={req.name}) and status=0`

表示查询是否已存在编码或名称等于要添加的产品信息的编码或名称，且未删除的产品。



设置好后，点确定按钮。

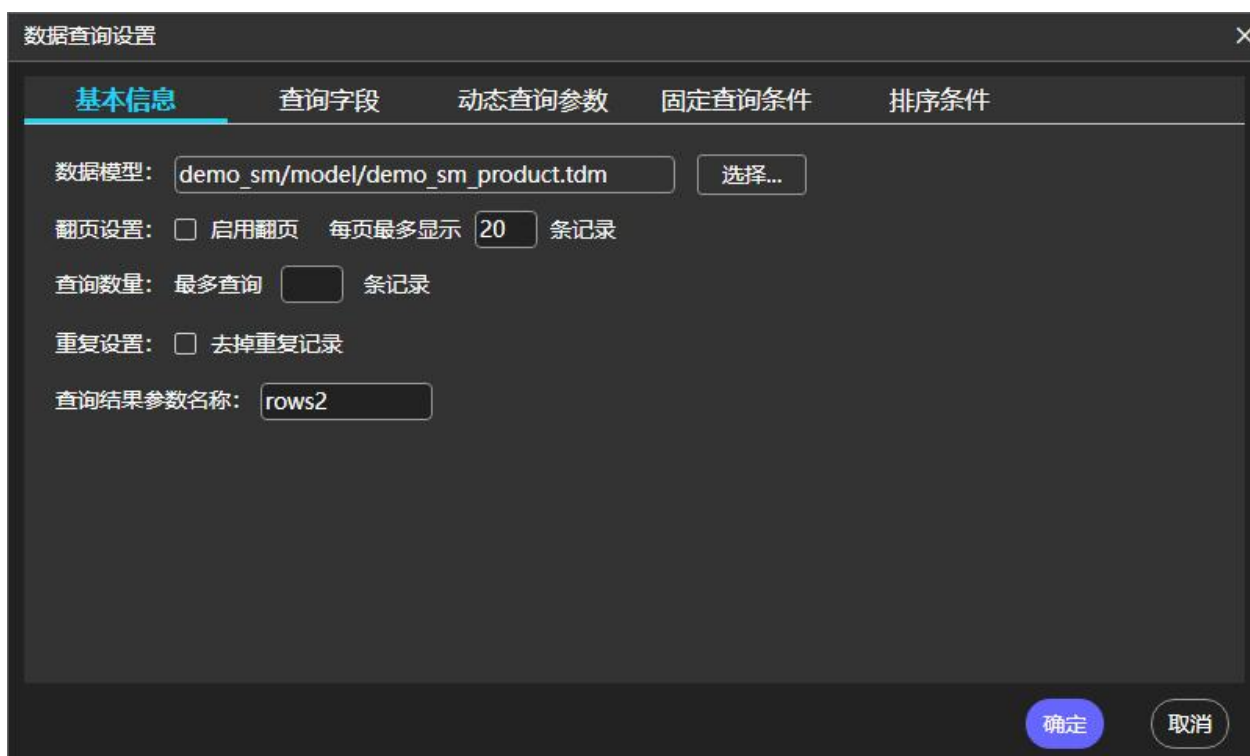
然后从基本语句里拖拽一个判断语句，判断条件设置为：`rows1.length>0`，表示如果查到编码或名称已经存在的产品信息，就执行判断语句内部的子语句。

再从基本语句里拖拽一个“返回逻辑错误”语句到上面的判断语句里，错误提示设置为“产品编码和产品名称不能重复！”，表示查询到有重复数据时，就提示这个错误信息。



#### 4.9.5.2 查询和设置产品序号

从数据库操作语句里拖拽一条数据查询语句到上面的数据查询语句里面、判断语句的下面，然后点击该语句的设置按钮，打开数据查询设置对话框，先在基本信息里选择数据模型：`demo_sm_product.tdm`，然后将查询结果参数名称设置为：`rows2`，如下图：



然后点击“查询字段”选项卡，选中字段 `sort_num`，字段格式设置为：`ifnull(max(sort_num), 0)` + 1，字段别名设置为：`sort_num`，表示查询最大的产品序号（如果为空，就设置为 0）并加 1：



然后点击“固定查询条件”选项卡，在此设置查询条件：`type_id={req.type_id} and status=0`，表示仅查询页面里选择的产品分类里状态正常的产品。



然后点确定，然后从基本语句里拖拽一条变量声明语句，变量名称设置为：maxSortNum，初始值设置为：rows2[0].sort\_num，如下图所示：



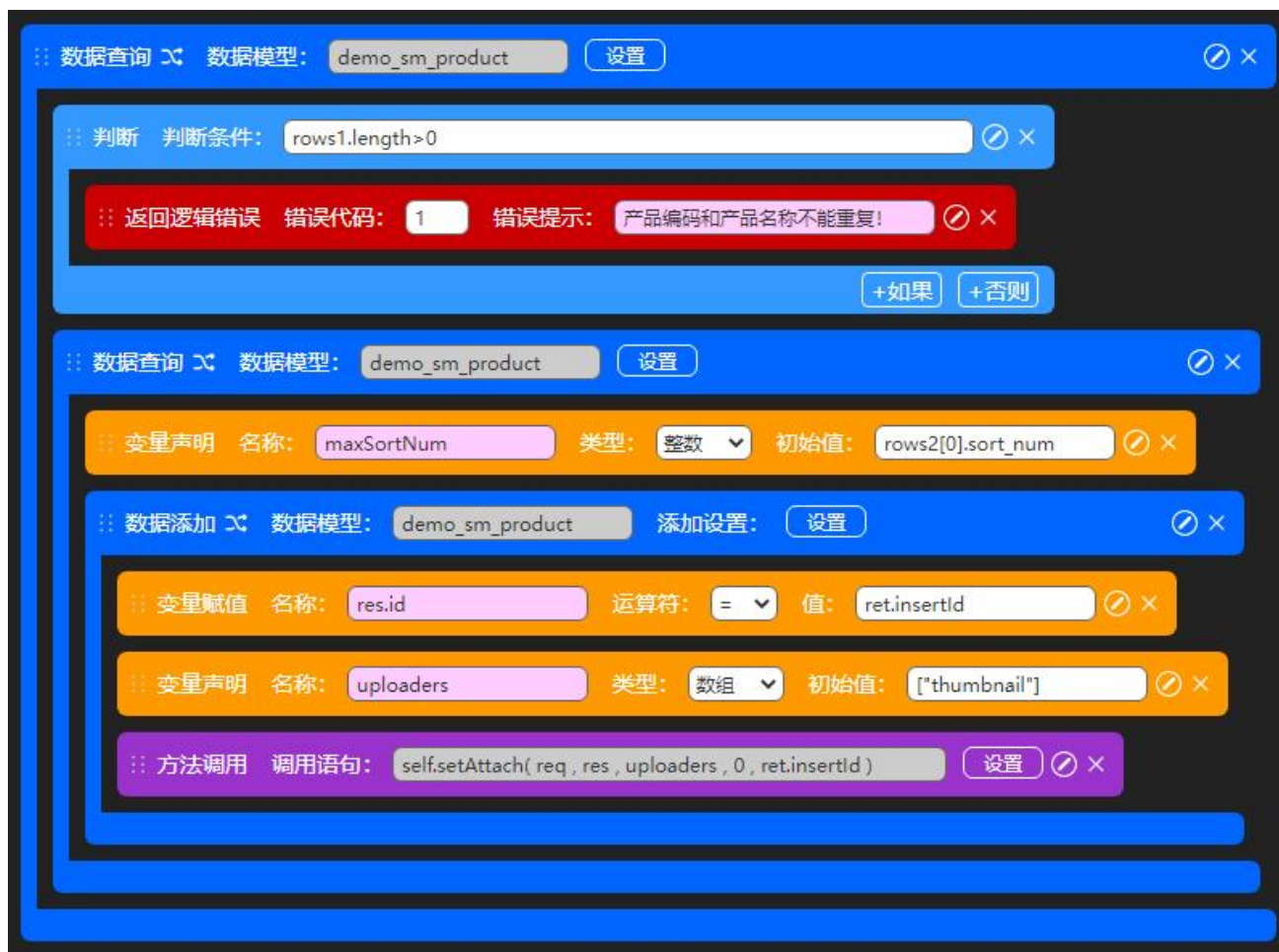
#### 4.9.5.3 修改数据添加语句

将原来用向导创建的数据添加语句拖拽到上面的查询最大产品序号的查询语句里、变量声明语句的下面，然后点击数据添加语句的“设置”按钮，将 sort\_num 字段的值设置为上面查询的现有产品的最大序号：maxSortNum，然后再将 thumbnail 字段的值修改成：req.thumbnail[0].code，表示在添加产品信息时，将产品信息缩略图字段 thumbnail 的值设置为页面里图片上传组件的第一个附件的附件代码。





点确定按钮后，最终修改完的后台服务界面如下图所示：

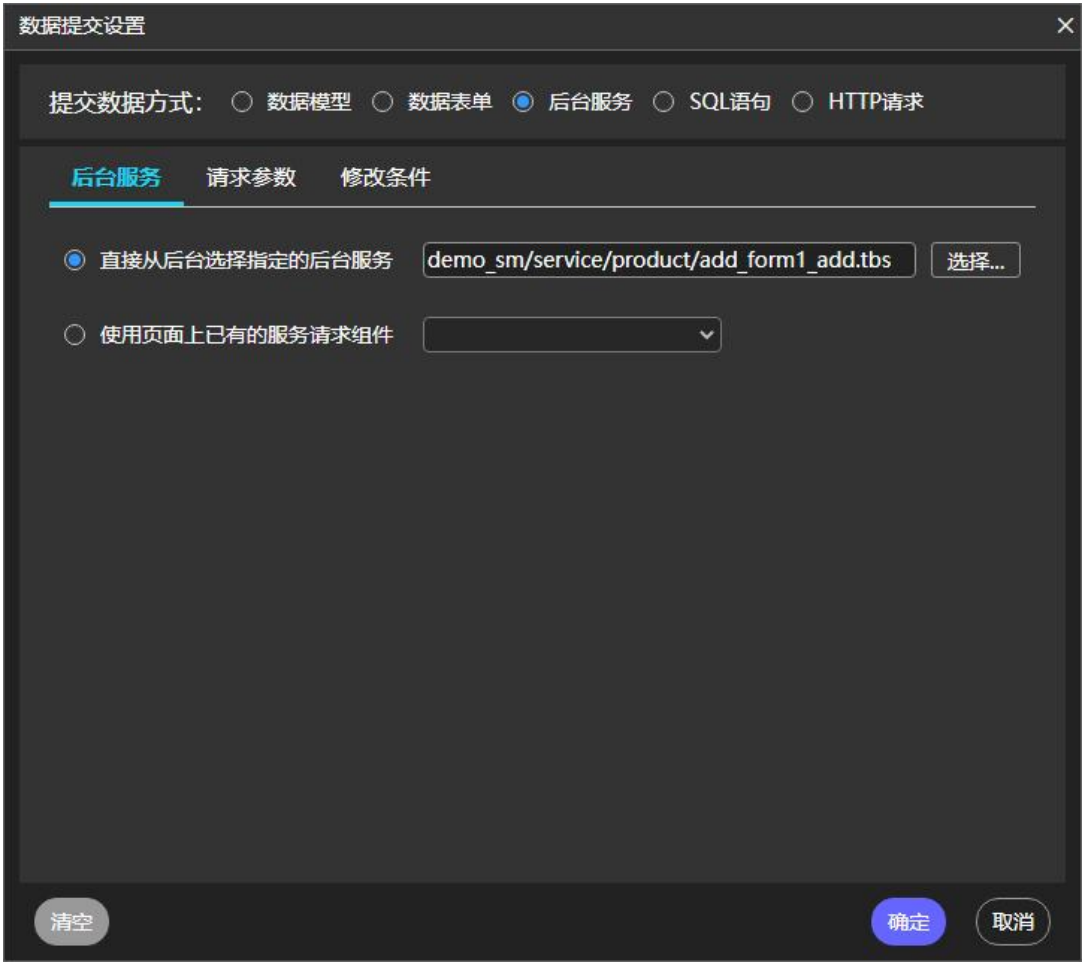


#### 4.9.5.4 修改表单提交数据的方法

接下来还需要修改添加产品信息页面里表单提交数据的方式，操作方法如下：

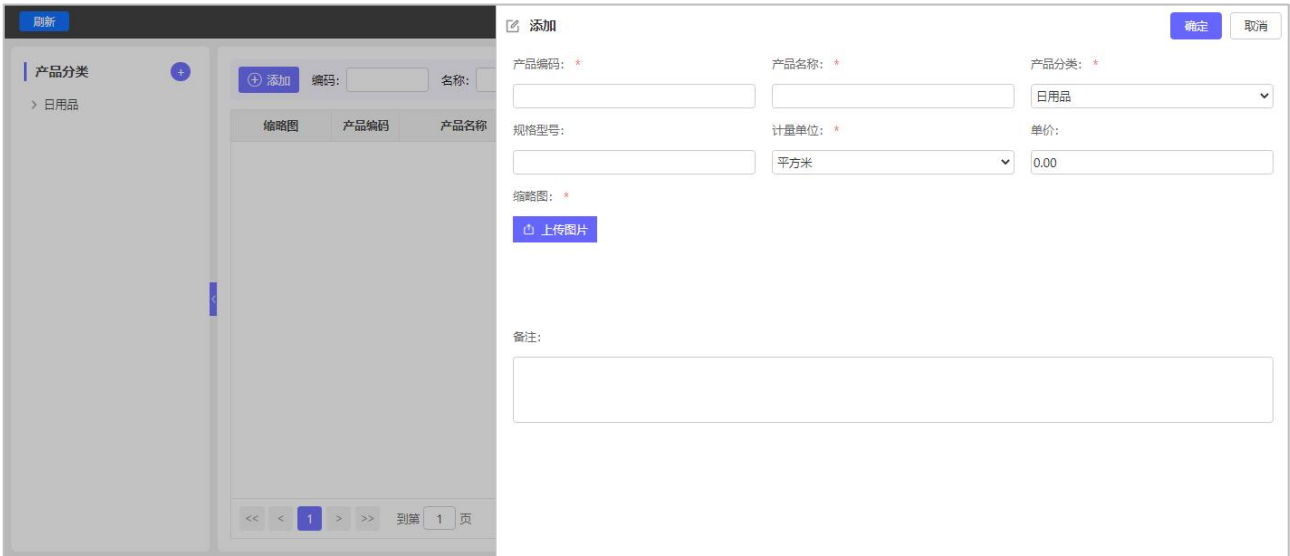
在添加产品信息页面里，选中表单组件 `form1`，然后点击属性里的“数据提交设置”，打开如下所示对话框，在里面先将“提交数据方式”修改为“后台服务”，然后选择前面修改的这个后台服务：`demo_sm/service/product/add_form1_add.tbs`，然后点击确定按钮，然后保存页面，这样设置后，后续如果再修改数据提交设置，就不会再重新自动生成这个后台服务，造成我们前面所做的修改被覆盖。



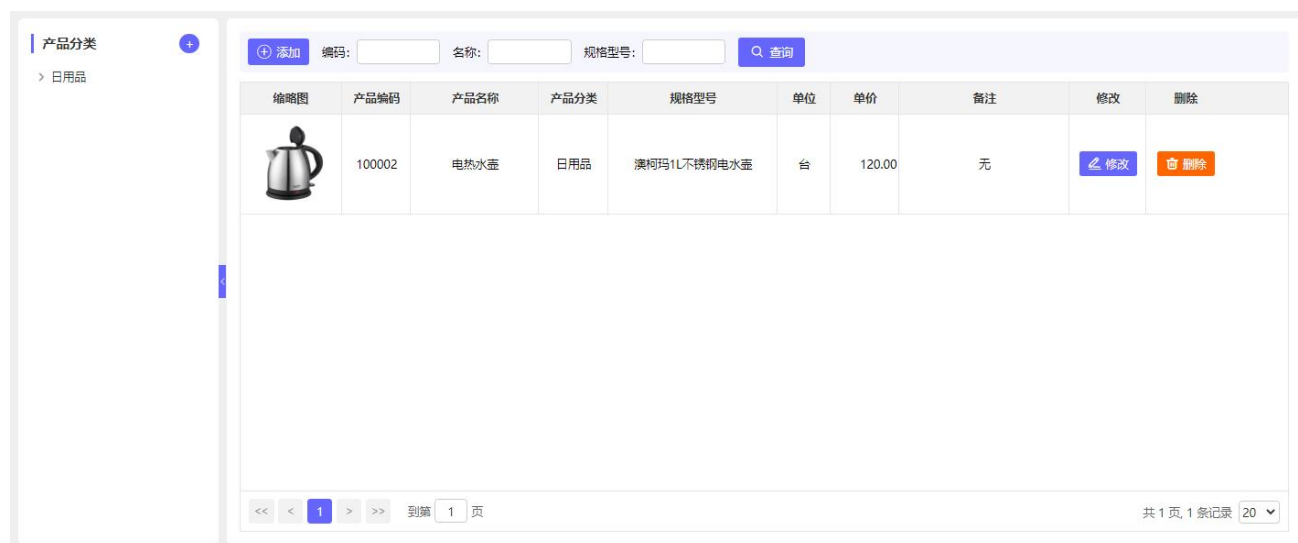


4.9.6 预览测试添加产品信息页面

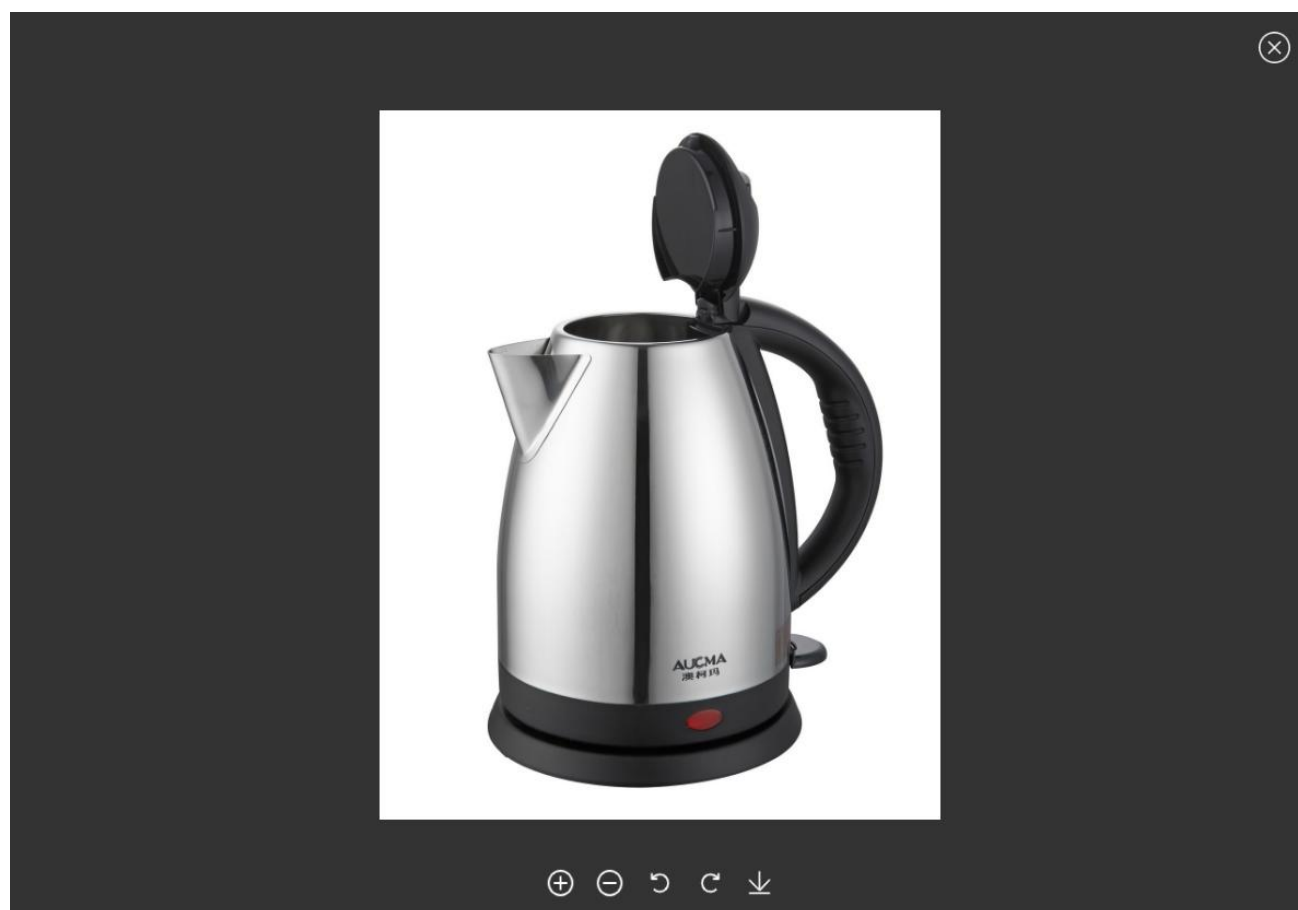
在 TaskBuilder 内打开产品信息列表页面，点击工具栏上的预览按钮，打开该页面的预览窗口，然后点击产品信息列表上面的添加按钮，就会打开添加产品信息的抽屉页，如下图所示：



在此输入产品信息，然后点击确定按钮，就可以看到列表里增加了新的产品信息。



点击产品缩略图可以打开图片查看器查看产品大图：



## 4.10 实现修改产品信息的功能

### 4.10.1 设置打开修改产品信息抽屉页的按钮

在数据表格里点击某一行数据里的“修改”按钮，即可打开修改产品信息的抽屉页，该按钮的设置方法如下：

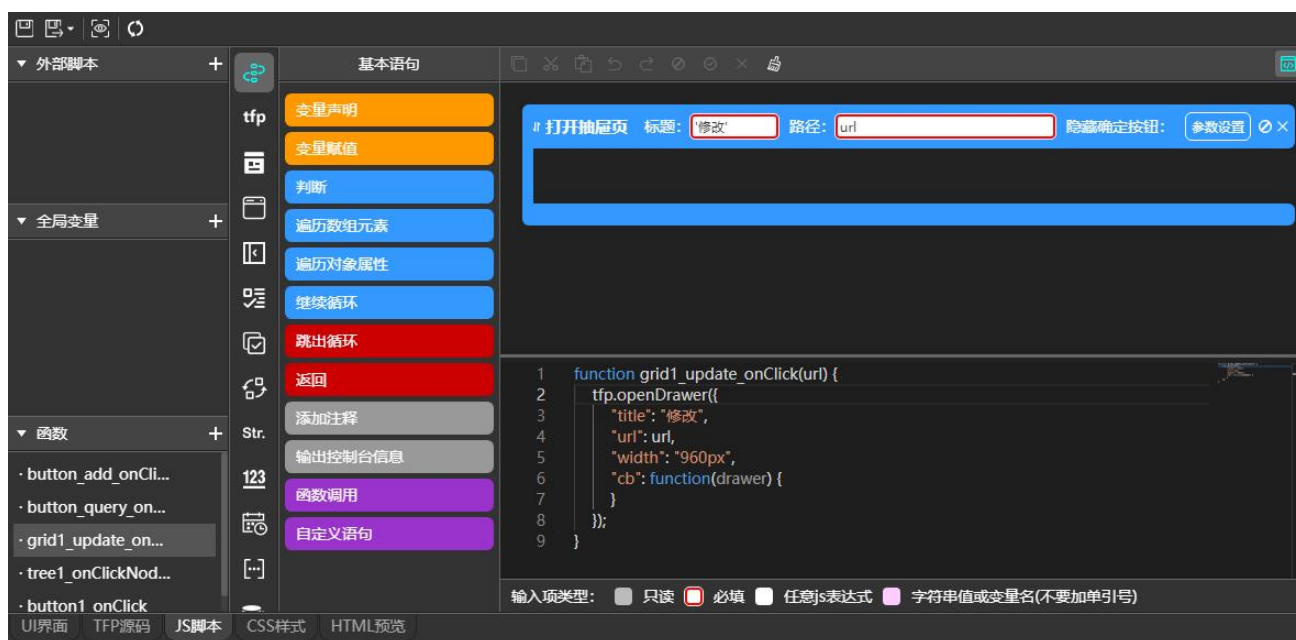
在产品信息列表页面里，选中数据表格组件，然后点击属性里的“数据列设置”，打开如下对话框，在左侧列表里选中“修改”这一列，可以看到，这一列的内容类型设置为了“按钮”，数据绑定格式设置为了固定的字符串：修改，点击后执行的脚本设置为了下面的脚本：

```
grid1_update_onClick('update.tfp?id={id}')
```

表示点击每一行的修改按钮时，会执行名为 `grid1_update_onClick()` 这个脚本函数，并传入修改页面的地址和当前产品信息的编号。



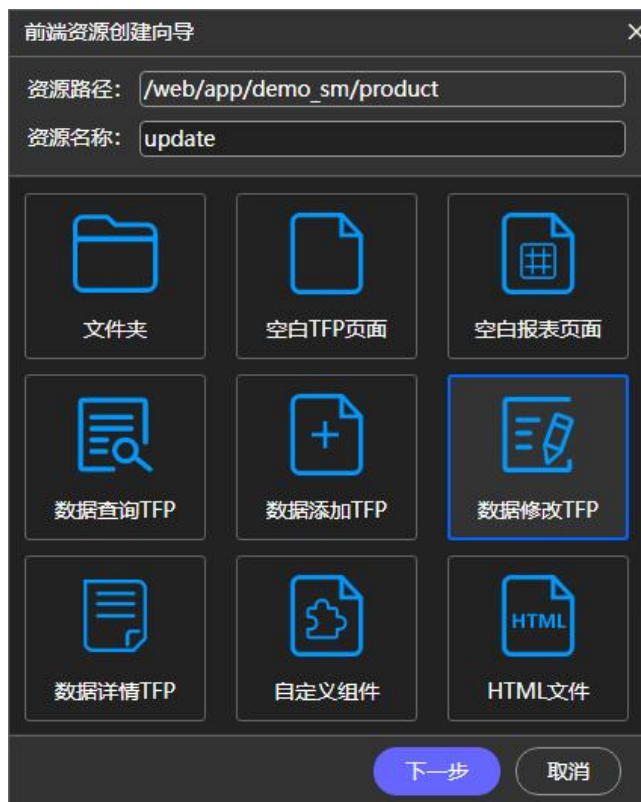
点取消按钮关闭该对话框，然后点设计器底部的“JS 脚本”，在左侧函数列表里选中 `grid1_update_onClick()` 函数，可以看到该脚本的设置界面，如下图所示：



在该函数里，就一条语句，就是打开修改产品信息的抽屉页。

## 4.10.2 创建修改产品信息的抽屉页

在销售管理示例项目的前端文件里，点击 product 目录右侧的加号，打开如下所示的“创建前端资源向导”对话框，在下面的资源类型列表中选择“数据修改 TFP”：



然后点下一步，打开如下所示的“创建数据修改 TFP”对话框，在此先选择数据模型：

demo\_sm\_product.tdm，然后设置页面标题：修改产品信息，然后再在下面的字段列表里按照下表选中和设置要修改的字段信息：

| 字段名       | 输入项名称 | 输入项类型 | 列数 | 必填 | 数据字典 | 默认值  |
|-----------|-------|-------|----|----|------|------|
| code      | 产品编码  | 单行输入项 |    | 是  |      |      |
| name      | 产品名称  | 单行输入项 | 2  | 是  |      |      |
| type_id   | 产品分类  | 下拉列表  |    | 是  |      |      |
| model     | 规格型号  | 单行输入项 | 2  |    |      |      |
| sort_num  | 产品序号  | 单行输入项 |    | 是  |      |      |
| unit      | 计量单位  | 单行输入项 |    | 是  | unit |      |
| price     | 单价    | 单行输入项 |    |    |      | 0.00 |
| thumbnail | 缩略图   | 图片上传  | 3  |    |      |      |
| comment   | 备注    | 多行输入框 | 3  |    |      |      |

与添加产品信息页面对比，多了一个产品序号输入框，以便在修改产品信息时，可以修改产品的序号，为了让抽屉页面更加美观，将产品名称和规格型号这两个输入项的宽度设置为占用 2 个正常输入项的位置。

创建数据修改TFP

保存路径: /web/app/demo\_sm/product/update.tfp 数据模型: demo\_sm/model/demo\_sm\_product.tdm 选择...

页面标题: 修改产品信息 页面类型: 抽屉页 窗口大小: 宽 高

终端类型: 电脑端 背景颜色: 浅色 定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧 每行显示输入项数量: 3 仅支持PC

数据查询服务: ☒ 自动创建 ☐ 使用已有服务 响应参数名: data 选择...

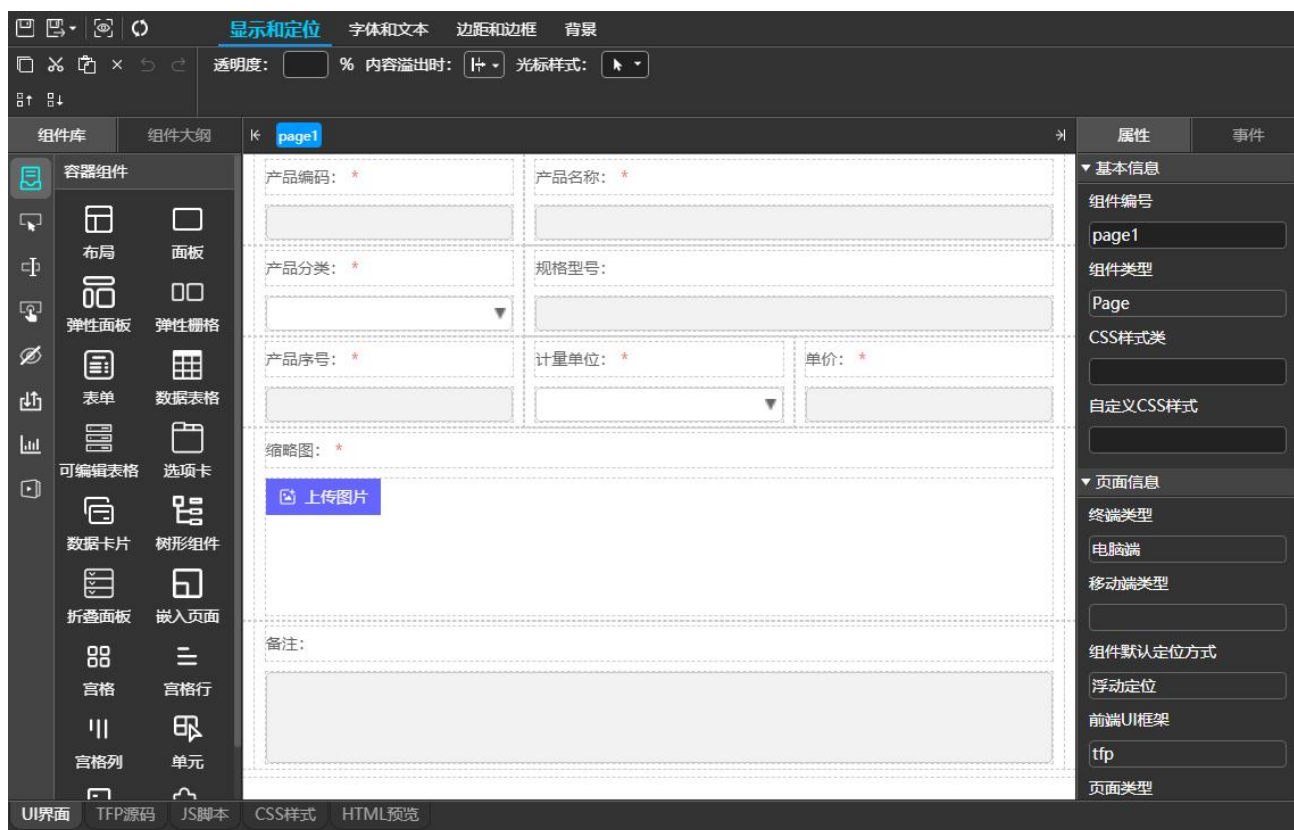
数据修改服务: ☒ 自动创建 ☐ 使用已有服务 选择...

前端页面设置字段 后台服务设置字段

| 字段名称                                         | 输入项名称 | 输入项类型 | 数据绑定格式     | 列数 | 必填                                  | 可选值  | 数据字典(双击选择) |
|----------------------------------------------|-------|-------|------------|----|-------------------------------------|------|------------|
| <input checked="" type="checkbox"/> code     | 产品编码  | 单行输入框 | {code}     |    | <input checked="" type="checkbox"/> |      |            |
| <input checked="" type="checkbox"/> name     | 产品名称  | 单行输入框 | {name}     | 2  | <input checked="" type="checkbox"/> |      |            |
| <input checked="" type="checkbox"/> type_id  | 产品分类  | 下拉列表  | {type_id}  |    | <input checked="" type="checkbox"/> |      |            |
| <input checked="" type="checkbox"/> model    | 规格型号  | 单行输入框 | {model}    | 2  | <input type="checkbox"/>            |      |            |
| <input checked="" type="checkbox"/> sort_num | 产品序号  | 单行输入框 | {sort_num} |    | <input checked="" type="checkbox"/> |      |            |
| <input checked="" type="checkbox"/> unit     | 计量单位  | 单行输入框 | {unit}     |    | <input checked="" type="checkbox"/> | unit |            |
| <input checked="" type="checkbox"/> price    | 单价    | 单行输入框 | {price}    |    | <input checked="" type="checkbox"/> |      |            |
| <input checked="" type="checkbox"/>          | 缩略图   | 图片上传  |            | 3  | <input checked="" type="checkbox"/> |      |            |
| <input checked="" type="checkbox"/>          | 备注    | 多行输入框 |            | 3  | <input checked="" type="checkbox"/> |      |            |

确定 取消

都设置好之后，点确定按钮完成创建，然后就可以在 TaskBuilder 里看到该页面：



### 4.10.3 完善修改产品信息抽屉页

与添加产品信息页面一样，用向导创建的修改产品信息页面需要继续完善才能使用，需要设置产品分类下拉列表组件和缩略图上传组件的相关参数，具体可以参考添加产品页面里的设置，这里就不再复述。

这里需要特殊说明的就是缩略图上传组件的数据绑定，在前面创建产品信息的数据模型时，我们定义了一个专门的字段 thumbnail，用来保存缩略图的附件代码，以便在产品列表里可以很方便地显示缩略图，在添加产品信息时，我们也修改了对应的后台服务，将上传的缩略图的附件代码存到了这个字段里，所以，在修改页面里，就可以用这个字段的值直接实现缩略图上传组件的数据绑定，只需要将其数据绑定格式设置为：{thumbnail} 即可，页面打开时，该组件会自动根据这里绑定的附件代码查询对应的缩略图信息。





像上面这种一个产品就一张缩略图的情况, 可以将这一张缩略图的附件代码保存到业务表的某个字段里, 然后将数据绑定格式设置为这个: {附件代码字段名} 即可。如果某个业务数据可能对应多个附件时, 则可以通过设置应用代码和应用数据 ID 这两个属性来实现自动绑定:



其中, 应用代码是由开发者自己随意设置的一个字符串, 只要保证每个应用里每一类附件的应用代码别重复就行, 例如如果产品信息里既有缩略图上传组件, 又有产品文档上传组件, 则这两个组件的应用代码建议分别设置为: product\_thumbnail 和 product\_document, 这样既能与其他应用里的附件进行区分, 又能区分同一个应用里的不同类型的附件。应用数据 ID 是指当前页面要修改的业务数据的主键值, 例如这里就是产品的 id。这两个属性设置好后, 打开页面时, 该组件就会根据这两个属性的值自动去系统附件表里查询该业务数据关联的附件信息, 并绑定到当前图片上传或文件上传组件上。

图片上传和文件上传组件都支持上述两种数据绑定方式, 但不要同时使用这两种方式。

#### 4.10.4 完善修改产品信息的后台服务

用向导自动创建的修改产品信息的后台服务存在一些问题, 需要完善之后才能使用。

在 TaskBuilder 内打开销售管理示例项目后台服务里 product 目录里的 update\_form1\_update.tbs, 然后进行以下操作:

#### 4.10.4.1 判断产品编码或名称是否已存在

在语句库的数据库操作中,拖拽一个数据查询语句到最上面,然后点击该语句的设置按钮,先在基本信息里选择数据模型 demo\_sm\_product.tdm, 如下图:

数据查询设置

基本信息 查询字段 动态查询参数 固定查询条件 排序条件

数据模型: demo\_sm/model/demo\_sm\_product.tdm 选择...

翻页设置: ☐ 启用翻页 每页最多显示 20 条记录

查询数量: 最多查询 条记录

重复设置: ☐ 去掉重复记录

查询结果参数名称: rows

确定 取消

然后点击“查询字段”选项卡,选中 id 字段:

数据查询设置

基本信息 查询字段 动态查询参数 固定查询条件 排序条件

添加自定义字段 添加关联字段

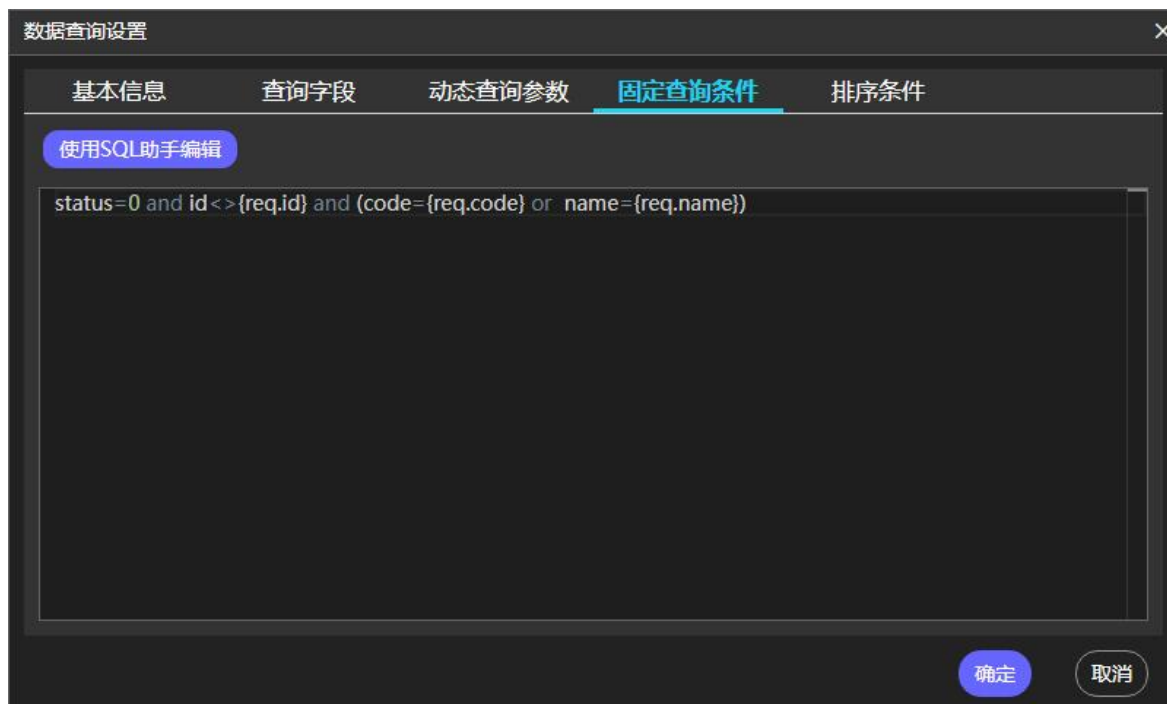
|                                     | 字段名称      | 字段备注   | 字段格式(SQL, 双击可编辑) | 字段别名 | 汇总 |
|-------------------------------------|-----------|--------|------------------|------|----|
| <input checked="" type="checkbox"/> | id        | 产品编号   |                  |      | ▼  |
| <input type="checkbox"/>            | code      | 产品编码   |                  |      | ▼  |
| <input type="checkbox"/>            | name      | 产品名称   |                  |      | ▼  |
| <input type="checkbox"/>            | type_id   | 产品分类编号 |                  |      | ▼  |
| <input type="checkbox"/>            | sort_num  | 产品序号   |                  |      | ▼  |
| <input type="checkbox"/>            | model     | 规格型号   |                  |      | ▼  |
| <input type="checkbox"/>            | unit      | 计量单位   |                  |      | ▼  |
| <input type="checkbox"/>            | price     | 单价     |                  |      | ▼  |
| <input type="checkbox"/>            | thumbnail | 缩略图代码  |                  |      | ▼  |
| <input type="checkbox"/>            | comment   | 备注     |                  |      | ▼  |

确定 取消

然后点击“固定查询条件”选项卡，在查询条件输入框里输入以下查询条件：

```
status=0 and id<>{req.id} and (code={req.code} or name={req.name})
```

表示查询状态正常，且编号不等于当前要修改的产品的编号，并且产品编码或名称等于页面里输入的编码和名称的产品信息。



设置好后，点确定按钮。

然后从基本语句里拖拽一个判断语句，判断条件设置为：`rows1.length>0`，表示如果查到编码或名称已经存在的产品信息，就执行判断语句内部的子语句。

再从基本语句里拖拽一个“返回逻辑错误”语句到上面的判断语句里，错误提示设置为“产品编码和产品名称不能重复！”，表示查询到有重复数据时，就提示这个错误信息。



#### 4.10.4.2 修改数据修改语句

将原来用向导创建的数据修改语句拖拽到上面的查询是否有重复数据的查询语句里、判断语句的下面，然后点击数据修改语句的“设置”按钮，将 thumbnail 字段的值修改成：`req.thumbnail[0].code`，表示在修改产品信息时，将产品信息缩略图字段 thumbnail 的值设置为页面里图片上传组件的第一个附件的附件代码。



数据修改设置对话框，显示了数据模型选择、字段映射表、修改条件（SQL）以及确定/取消按钮。

数据模型:  选择...

|                                     | 字段名称      | 字段备注   | 字段值 (js表达式)           |
|-------------------------------------|-----------|--------|-----------------------|
| <input checked="" type="checkbox"/> | type_id   | 产品分类编号 | req.type_id           |
| <input checked="" type="checkbox"/> | sort_num  | 产品序号   | req.sort_num          |
| <input checked="" type="checkbox"/> | model     | 规格型号   | req.model             |
| <input checked="" type="checkbox"/> | unit      | 计量单位   | req.unit              |
| <input checked="" type="checkbox"/> | price     | 单价     | req.price             |
| <input checked="" type="checkbox"/> | thumbnail | 缩略图代码  | req.thumbnail[0].code |
| <input checked="" type="checkbox"/> | comment   | 备注     | req.comment           |

修改条件(SQL): 使用SQL助手编辑

```
id={req.id}
```

确定 取消

点确定按钮后，最终修改完的后台服务界面如下图所示：



#### 4.10.4.3 修改表单提交数据的方法

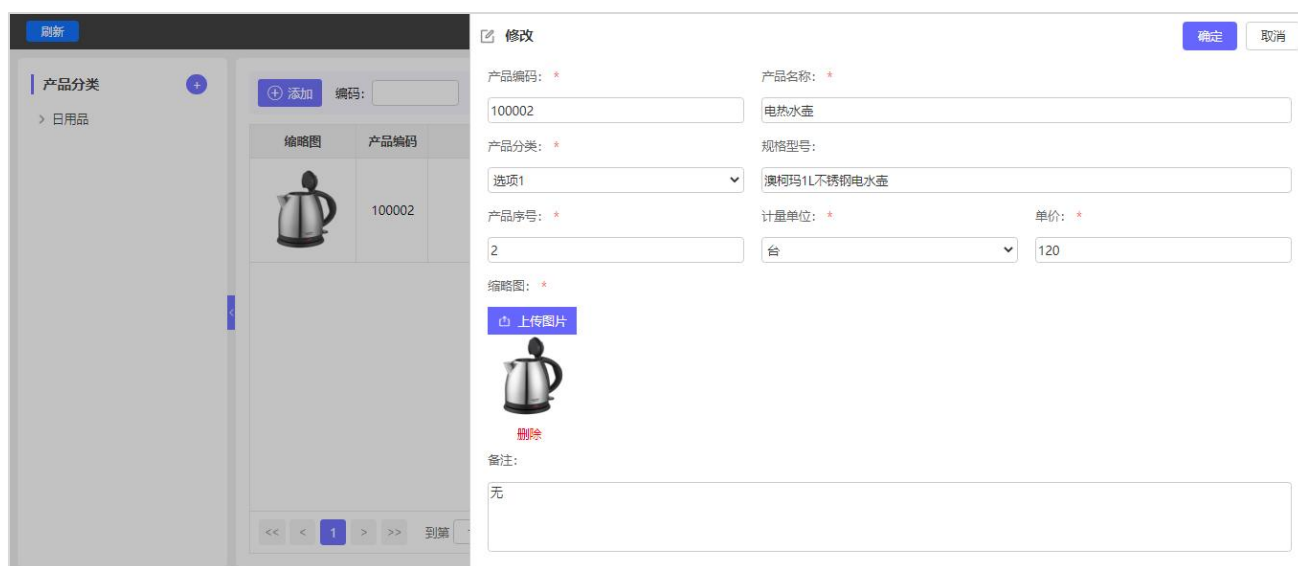
接下来还需要修改“修改产品信息”页面里表单提交数据的方式，操作方法如下：



在修改产品信息页面里，选中表单组件 form1，然后点击属性里的“数据提交设置”，打开如下所示对话框，在里面先将“提交数据方式”修改为“后台服务”，然后选择前面修改的这个后台服务：demo\_sm/service/product/update\_form1\_update.tbs，然后点击确定按钮，然后保存页面，这样设置后，后续如果再修改这个页面的数据库操作有关的设置，就不会再重新自动生成这个后台服务，造成我们前面所做的修改被覆盖。

### 4.10.5 预览测试修改产品信息的抽屉页

在 TaskBuilder 内打开产品信息列表页面，点击工具栏上的预览按钮，打开该页面的预览窗口，然后点击产品信息列表里某个产品右边的修改按钮，就会打开该产品的修改页面，如下图所示：



The screenshot shows a '修改' (Edit) form for a product. The form is divided into two main sections: a left sidebar and a main right area. The sidebar contains a list of products with columns for '缩略图' (Thumbnail) and '产品编码' (Product Code). The main area contains various input fields for '产品编码', '产品名称', '产品分类', '规格型号', '产品序号', '计量单位', '单价', and a '缩略图' upload section. There are also '确定' (Confirm) and '取消' (Cancel) buttons at the top right.

在此修改产品信息，然后点击确定按钮，就可以看到列表里该产品的信息发生了变化。

## 4.11 完善产品信息的删除功能

在前面用向导创建的产品信息列表页面里，点击某一行产品数据最后面的删除按钮时，会将该行数据记录直接从数据库中删除，这对于产品这种重要信息来说，不太妥当，万一操作失误，要再找回就比较麻烦。建议改成逻辑删除，即在删除时，不真的删除该条数据记录，只是



将其状态修改为“已删除”，并且记录删除人和删除时间，以便后续查证。具体操作方法如下：

在产品信息列表页面里，选中数据表格组件，然后点击属性里的“数据删除设置”，在打开的对话框中，在下面的“删除方式”中，选择“逻辑删除”，并在下面的字段列表设置要修改的三个字段及要修改的值：

- **状态**：不选择任何系统宏变量，直接添加，然后设置为-1，表示已删除。
- **删除人编号**：在系统宏变量里选择当前用户编号，对应的 JS 表达式为：req.session.userId；
- **删除时间**：在系统宏变量里选择当前日期时间，对应的 JS 表达式为：new Date()；

设置完后的界面如下图所示，然后点确定，保存列表页面，再删除就是逻辑删除了：

数据删除设置

删除数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型

请求参数

☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

删除条件 可选字段： 判断条件：

id={req.id}

删除方式：☐ 物理删除 ☒ 逻辑删除

字段列表： 系统宏变量：

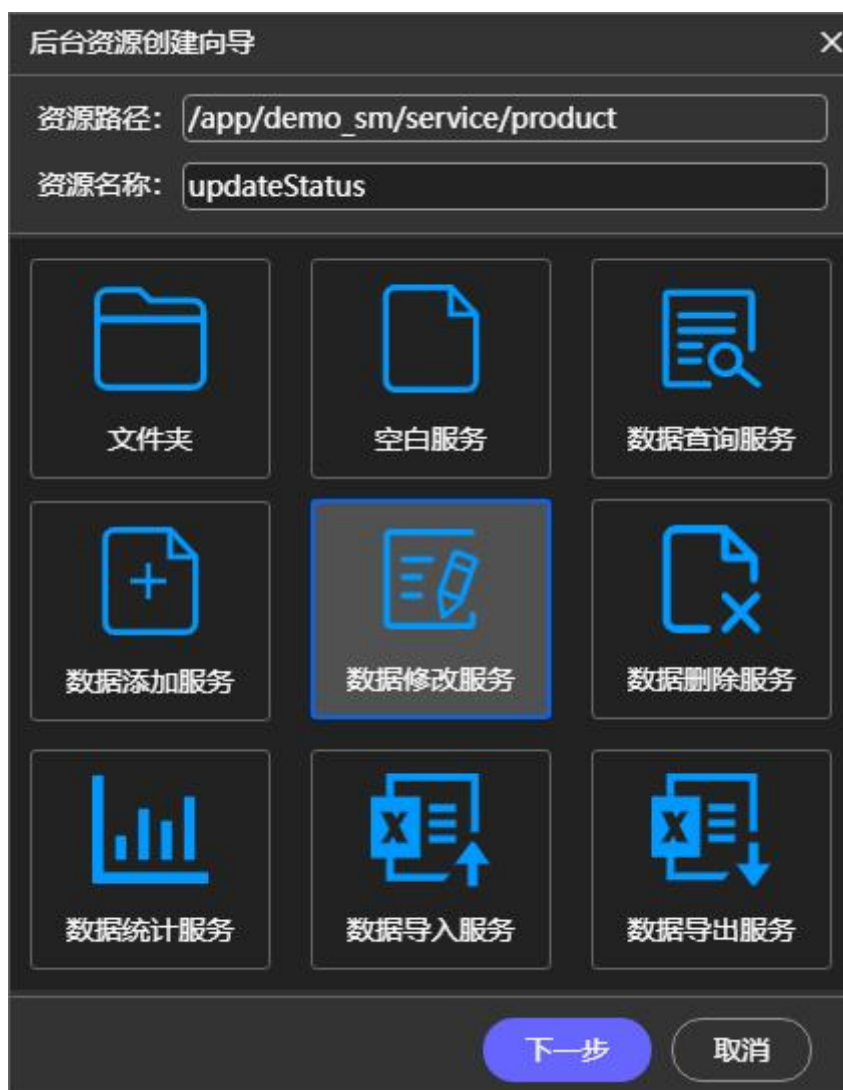
| 字段名称        | 字段备注  | 字段值                |                                                                  |
|-------------|-------|--------------------|------------------------------------------------------------------|
| status      | 状态    | -1                 | <input data-bbox="1289 1704 1326 1749" type="button" value="-"/> |
| deletor_id  | 删除人编号 | req.session.userId | <input data-bbox="1289 1760 1326 1805" type="button" value="-"/> |
| delete_time | 删除时间  | new Date()         | <input data-bbox="1289 1816 1326 1861" type="button" value="-"/> |

## 4.12 实现产品信息禁用和启用

新添加的产品，状态默认都是正常的（status=0），也就是已启用的，有时候有些产品可能暂时不允许对外销售，这时就可以将其状态改为已禁用（status=1），等可以销售了，再改为正常。下面教大家具体怎么来实现。

### 4.12.1 创建修改产品信息状态的后台服务

在 TaskBuilder 里，点击销售管理示例项目的后台服务里的 product 目录右侧的加号，打开如下图所示的“后台资源创建向导”，选中“数据修改服务”，资源名称设置为：updateStatus。



然后点击下一步，弹出如下所示的“创建数据修改服务”对话框：

创建数据修改服务

服务路径: /app/demo\_sm/service/product/updateStatus.tbs

服务说明: 修改产品状态 编辑模式: ☒ 图形化模式 ☐ 代码模式

数据模型: demo\_sm/model/demo\_sm\_product.tdm 选择...

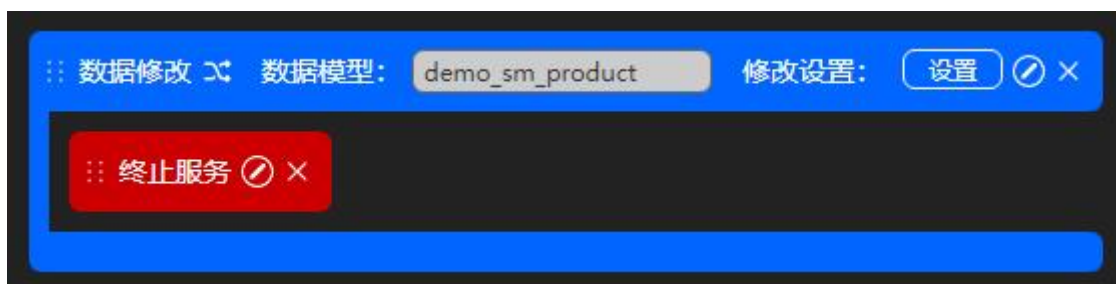
| <input type="checkbox"/>            | 字段名称        | 字段备注  | 字段值(js表达式)      | 不为空                      |
|-------------------------------------|-------------|-------|-----------------|--------------------------|
| <input type="checkbox"/>            | price       | 单价    | req.price       | <input type="checkbox"/> |
| <input type="checkbox"/>            | thumbnail   | 缩略图代码 | req.thumbnail   | <input type="checkbox"/> |
| <input type="checkbox"/>            | comment     | 备注    | req.comment     | <input type="checkbox"/> |
| <input checked="" type="checkbox"/> | status      | 状态    | req.status      | <input type="checkbox"/> |
| <input type="checkbox"/>            | deletor_id  | 删除人编号 | req.deletor_id  | <input type="checkbox"/> |
| <input type="checkbox"/>            | delete_time | 删除时间  | req.delete_time | <input type="checkbox"/> |

修改条件(SQL): 使用SQL助手编辑

```
id in (${req.id})
```

确定 取消

在此，先选中数据模型 demo\_sm\_product.tdm，服务说明设置为：修改产品状态，然后在下面的字段列表中选中字段 status，然后将修改条件设置为：id in (\${req.id})，表示修改条件为：id 字段值等于前端页面传过来的一个或多个 id 值，多个 id 值用英文逗号分隔，然后点确定按钮即可完成创建，创建后的界面如下图所示：



该服务还需要修改一下才能使用，在 TaskBuilder 里打开该服务后，点击底部的“基本信息”，可以看到如下图所示界面：

保存 另存为 另存到本地 测试 刷新

服务名称:

服务说明:

安全设置: ☐ 不校验是否登录 ☐ 不检查SQL注入

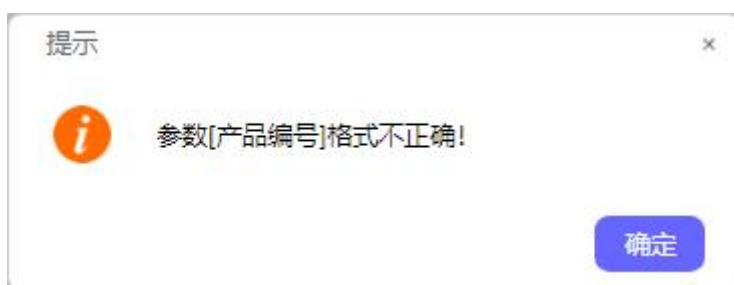
请求参数: +

| 参数名    | 参数备注 | 参数类型 | 不为空                                 | 默认值                  | 忽略注入                     | 文本验证                                |
|--------|------|------|-------------------------------------|----------------------|--------------------------|-------------------------------------|
| status | 状态   | 整数   | <input checked="" type="checkbox"/> | <input type="text"/> | <input type="checkbox"/> | <input type="text"/> <span>⊖</span> |
| id     | 产品编号 | 字符串  | <input checked="" type="checkbox"/> | <input type="text"/> | <input type="checkbox"/> | <input type="text"/> <span>⊖</span> |

基本信息 设计器 TBS代码 JS代码

在上图的请求参数列表里，需要进行以下修改：

- **status**：将参数类型设置为整数，不为空；
- **id**：将参数类型设置为字符串，因为前端页面有可能传多个 id 值，这时多个 id 值之间用逗号分隔，如果用默认的整数类型，则请求该服务时就会收到如下图所示的错误提示：



#### 4.12.2 在 SQL 中插入 JS 变量的两种方式

这里涉及到一个知识点，就是在后台服务的 SQL 语句或表达式里，有以下两种方式可以传入后台 JavaScript 的变量值：

- **{JS 变量}**：这种形式只能作为一个具体的值传递给要执行的 SQL 语句，例如在添加或修改

语句里，作为某个字段的值，查询语句里，作为查询条件里某个字段的值等，不能将其作为 SQL 关键字或字段名拼接到 SQL 语句里；采用这种形式传值，JS 变量的值不是直接拼接到 SQL 语句里，而是传给数据库管理软件，在实际执行 SQL 语句时，再作为参数传入，这样可以有效地防止 SQL 注入，即使攻击者在参数中传了一些试图攻击的特殊字符串，数据库管理软件也只会将其当作普通的字符串处理，而不会作为 SQL 语句执行。

- **`\${JS 变量}`**：这种形式是将 JS 变量的值直接拼接到 SQL 语句中，这种形式适合根据 JS 变量的值动态构建一条复杂的 SQL 语句，例如上面的查询条件：`id in (${req.id})`，就是将前端传过来的 id 参数直接拼接到这个查询条件中，这样的话，如果有多个 id 值，并且以英文逗号分隔，例如：“1,2,3”，拼接完的查询条件就是：`id in (1,2,3)`，这样就可以实现对多条记录的批量修改，而如果采用 `id={req.id}` 这种形式，就只能修改一条记录。这种方式的缺点就是存在安全隐患，攻击者可以在参数中传入特殊的字符串，实现 SQL 注入，所以，一般情况下，不建议采用该方式。

### 4.12.3 修改产品列表页面相关设置

要实现产品信息的启用和禁用，产品列表页面需要做如下修改：

#### 4.12.3.1 数据表格组件显示选择框

在产品列表页面内选中数据表格组件，然后在属性里，将“显示选择框”开关打开，则在预览窗口或浏览器中打开产品列表页时，产品列表前面就会出现选择框，以便选择多个产品。

#### 4.12.3.2 添加服务请求组件

从组件库的不可视组件中，拖拽“服务请求”组件到列表页面上，然后设置以下属性：

- **组件编号**：设置为 `serviceUpdateStatus`；
- **服务路径**：`demo_sm/service/product/updateStatus.tbs`；

### 4.12.3.3 添加和设置启用按钮

从组件库的只读组件中拖拽按钮组件到产品列表上面的查询面板中，放在查询按钮后面，并设置以下属性（如果屏幕分辨率小，查询面板会出现滚动条，需要滚动到最右侧才能看到）：

- **组件编号**：设置为 `buttonEnable`；
- **按钮文本**：设置为启用；
- **按钮类型**：设置为成功按钮；
- **显示图标**：将该选项的开关打开；
- **图标**：点击右侧图标按钮，打开图标选择对话框，选中“确定”图标；

然后在上面的样式设置栏里，在边距和边框里，将外边距里的左边距设置为 15，最终设置完的效果如下图：



然后双击事件设置里的“点击时”事件，打开事件脚本编辑器，在此添加以下语句：

- (1) 从语句库的组件操作中拖拽一条“调用组件方法”语句，组件设置为：`grid1`，方法设置为：获得选中行的主键值，该方法会返回一个变量，默认为：`keys`，里面保存的是在产品列表里选中的一个或多个产品的 `id` 值，多个 `id` 值用英文逗号分隔，如果没有选择任何产品则为空；
- (2) 从语句库的基本语句中拖拽一条“判断”语句，判断条件设置为：`!keys`，表示如果在产品列表里选中的产品的 `id` 值为空，就执行判断语句里的子语句；
- (3) 从语句库的“tfp 框架基本功能”中拖拽一条“显示弹窗提示”语句到判断语句里，将提示信息设置为：`'请选择要启用的产品！'`，**注意**：这里要使用英文单引号，不要用双引号；
- (4) 从语句库的基本语句中拖拽一条“返回”语句到“显示弹窗提示”语句的下面，表示提示后就退出该脚本函数，不再执行后续脚本；



(5) 从语句库的“tfp 框架基本功能”中拖拽一条“显示弹窗确认”语句到上面的判断语句下面，将提示信息设置为：'确定启用吗？'，表示点击启用按钮后先提示用户是否确定要启用；

(6) 从语句库的基本语句中拖拽一条“判断”语句到上面的“显示弹窗确认”语句的里面，判断条件设置为：ret，表示如果在确认窗口中点击了确定按钮，就执行判断语句里的子语句；

(7) 从语句库的组件操作中拖拽一条“调用组件方法”语句到上面的“判断”语句里，组件设置为：serviceUpdateStatus，方法设置为：设置请求参数，然后点参数设置，在弹出的对话框里按下图所示设置参数，表示在发送请求时，将 id 参数设置为前面第一条语句里返回的变量 keys，然后点确定保存：

| 参数名   | 参数备注 | 参数类型 | 必填 | 参数值  |
|-------|------|------|----|------|
| name  | 参数名称 | 字符串  | 是  | 'id' |
| value | 参数值  | 字符串  | 是  | keys |

(8) 从语句库的组件操作中拖拽一条“调用组件方法”语句到上面第 7 条语句下面，组件设置为：serviceUpdateStatus，方法设置为：设置请求参数，然后点参数设置，在弹出的对话框里按下图所示设置参数，表示在发送请求时，将 status 参数设置为 0，然后点确定保存：

| 参数名   | 参数备注 | 参数类型 | 必填 | 参数值      |
|-------|------|------|----|----------|
| name  | 参数名称 | 字符串  | 是  | 'status' |
| value | 参数值  | 字符串  | 是  | 0        |

(9) 然后再从语句库的组件操作中拖拽一条“调用组件方法”语句到上面第 8 条语句下面，组件还是设置为：serviceUpdateStatus，方法设置为：发送请求，可以看到该语句可以包含子语句，因为服务请求组件的发送请求方法在收到服务器响应结果后会执行回调函数，这些子语句就是放在回调函数里的；

(10) 然后再从语句库的组件操作中拖拽一条“调用组件方法”语句到上面第 9 条语句的里面，组件设置为：grid1，方法设置为重新加载数据，表示在修改完产品状态后，重新加载表格里的数据。

最终设置完的界面如下图所示：



#### 4.12.3.4 添加和设置禁用按钮

再拖拽一个按钮到页面里,用来禁用选中的产品信息,该按钮的设置和启用按钮基本一致,这里就不详细介绍了,主要说一下不一样的地方。

属性设置不一样的地方包括以下四个：

- **组件编号**：设置为 buttonDisable；
- **组件类型**：设置为危险按钮；
- **按钮文本**：设置为禁用；

- **图标：** 设置为取消；

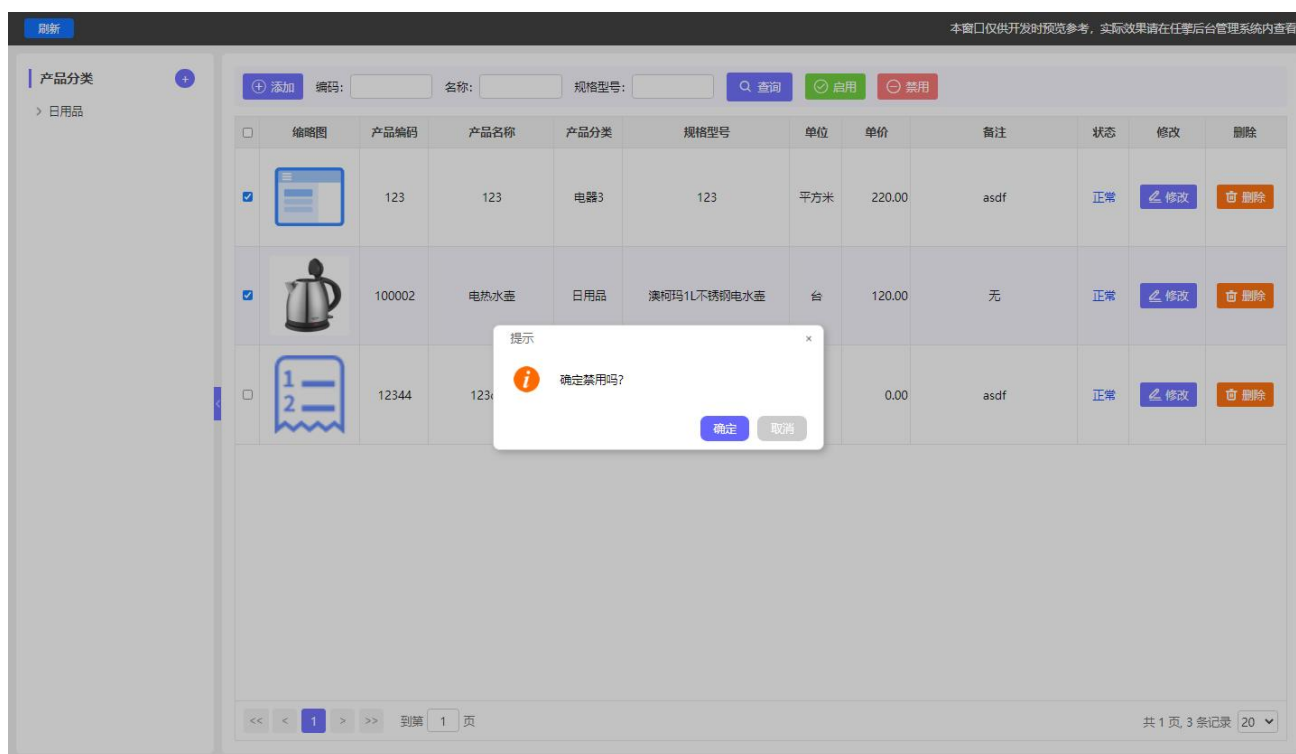
点击事件的脚本里，有以下几个地方不一样：

(1) 显示弹窗提示语句和显示弹窗确认语句里的“启用”改成“禁用”；

(2) 设置 serviceUpdateStatus 组件的第 2 个“设置请求参数”语句里(参数名称为 status)，将参数值改为 1；

这些都设置好后，就可以预览测试启用和禁用的功能了。

预览效果如下图所示：



## 4.13 实现产品信息导入和导出

接下来带领大家实现产品信息的导入和导出功能。

### 4.13.1 实现产品信息导入功能

#### 4.13.1.1 添加和设置 Excel 导入组件

在 TaskBuilder 里打开产品信息列表页面，从组件库的导入导出中拖拽“Excel 导入”组件

到产品列表上面的面板组件中，在上面的样式设置栏的边距和边框里，将外边距里的左边距设置为 15，然后点属性里的“数据导入设置”，打开如下所示对话框：

数据导入设置

提交数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务

数据模型

请求参数

修改条件

安全设置：☐ 不校验是否登录 ☐ 不检查SQL注入

标题行位置：在第  行

记录存在时：☒ 修改现有值 ☐ 删除重建 ☐ 不做任何处理

导入数据为空时：☐ 将现有数据也修改为空

数据模型路径：

选择...

对每行数据进行特殊处理的后台服务：

选择...

| <input type="checkbox"/>            | 字段名称     | 数据类型 | Excel列标题 | 必填                                  | 主键                                  | 计算表达式        | 默认值 | 可选值(英文逗号分隔) |
|-------------------------------------|----------|------|----------|-------------------------------------|-------------------------------------|--------------|-----|-------------|
| <input checked="" type="checkbox"/> | code     | 文本   | 产品编码     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |              |     |             |
| <input checked="" type="checkbox"/> | name     | 文本   | 产品名称     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |              |     |             |
| <input checked="" type="checkbox"/> | type_id  | 整数   | 产品分类     | <input type="checkbox"/>            | <input type="checkbox"/>            |              | 0   |             |
| <input checked="" type="checkbox"/> | sort_num | 整数   | 产品序号     | <input type="checkbox"/>            | <input type="checkbox"/>            | { rowIndex } |     |             |
| <input checked="" type="checkbox"/> | model    | 文本   | 规格型号     | <input type="checkbox"/>            | <input type="checkbox"/>            |              |     |             |
| <input checked="" type="checkbox"/> | unit     | 文本   | 计量单位     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |              |     |             |
| <input checked="" type="checkbox"/> | price    | 数字   | 单价       | <input type="checkbox"/>            | <input type="checkbox"/>            |              | 0   |             |

清空

确定

取消

在该对话框中，有以下参数可以设置：

- **标题行位置**：设置 Excel 里包含标题文字的行所在的位置，默认为第 1 行，这里默认即可；
- **记录存在时如何处理**：可以选修改现有值、删除重建、不做任何处理这三种方式，默认为修改现有值，这种情况至少需要将一个字段设置为主键，才能通过该字段的值判断导入的数据是否已经存在，这里默认即可；
- **导入数据为空时是否将现有数据也修改为空**：默认未选中，表示不修改，这里默认即可；
- **数据模型路径**：设置要导入的数据模型，可以点右侧的选择按钮进行选择，必须先选择数据模型，才能在下面的字段列表里选择要导入的字段，这里选择 demo\_sm\_product.tdm；
- **对每行数据进行特殊处理的后台服务**：如果要对每行导入的数据进行特殊处理或判断，则

可以在此设置处理数据的后台服务路径, 点击右侧的选择按钮可以选择, 这里先不用设置, 后面会介绍怎么用;

然后在下面的字段列表中, 可以选择要导入的字段, 并设置这些字段的以下参数:

- **Excel 列标题**: 用来设置 Excel 文件里当前字段对应的列标题文字;
- **必填**: 用来设置该字段的值是否是必填, 不能为空;
- **主键**: 用来设置哪个字段是主键, 以使用该字段的值判断数据库里是否已经有对应数据;
- **计算表达式**: 用来设置导入时在后台计算当前字段值的 JS 表达式, 可以是任意 JavaScript 值或表达式, 例如: 0、'abc'、new Date()、req.session.userId 等, 如果要根据导入的 Excel 里的数据进行拼接或计算, 则可以采用 {Excel 列标题} 的形式在 JS 表达式中插入指定列的值, 例如可以将备注设置为: **编码: {产品编码}, 名称: {产品名称}, 单位: {计量单位}**; 另外, 还可以使用一个特殊的变量: **{\_rowIndex}**, 对应导入数据的行索引, 从 0 开始; 如果设置了计算表达式, 则 Excel 内不需要提供该列的数据, 系统会自动生成;
- **默认值**: 用来设置如果在 Excel 内, 某一行该字段对应的列没有数据, 则将该字段的值设置为这里设置的默认值, 格式为任意 JavaScript 值或表达式, 例如: 0、'abc'、new Date()、req.session.userId 等。
- **可选值**: 用来设置该字段可选的选项值, 多个值用英文逗号分隔, 在执行导入时, 如果某一行该字段对应的列的值不属于这里设置的任意一个可选值, 则不导入该行数据;

对于我们要导入的产品信息, 请按照下表所示选择和设置要导入的字段信息:

| 字段名称     | Excel 列标题 | 必填 | 主键 | 计算表达式       | 默认值 |
|----------|-----------|----|----|-------------|-----|
| code     | 产品编码      | 是  | 是  |             |     |
| name     | 产品名称      | 是  |    |             |     |
| type_id  | 产品分类      |    |    |             | 0   |
| sort_num | 产品序号      |    |    | {_rowIndex} |     |
| model    | 规格型号      |    |    |             |     |

|         |      |   |  |  |   |
|---------|------|---|--|--|---|
| unit    | 计量单位 | 是 |  |  |   |
| price   | 单价   |   |  |  | 0 |
| comment | 备注   |   |  |  |   |
| status  | 状态   |   |  |  | 0 |

然后点确定保存设置，然后再保存列表页面，就可以预览测试导入功能了。

#### 4.13.1.2 添加和设置处理每行导入数据的后台服务

我们测试会发现有两个问题：

(1) 无法导入产品的分类信息，在产品信息的数据模型里，存的是分类编号，这个分类编号是没有任何业务含义的，导入时不可能让用户在 Excel 里输入分类编号，只能输入分类编码或名称，但这个字段又不能为空，所以上面先将分类编号都设置为了 0；

(2) 计量单位也存在同样的问题，在产品信息的数据模型里，存的是单位编码，导入时，也没办法要求用户在 Excel 里输入这个编码，只能输入名称；

为了解决以上两个问题，需要通过创建和设置一个专门处理每行导入的产品信息的后台服务来实现，在前面介绍“Excel 导入”组件的设置参数时，里面介绍了一个名为“对每行数据进行特殊处理的后台服务”的参数，这里如果设置了一个指定的后台服务，则在执行数据导入时，在导入每行数据之前，都会先调用这个后台服务，并把当前行的数据通过 req.rowData 传给这个服务，然后就可以在这个后台服务里进行以下操作：

(1) 根据导入的数据查询关联数据，并替换相应列的数据，就例如上面说的导入分类编号的问题，就可以在导入时先让用户在 Excel 的“产品分类”这一列里输入分类名称，然后在这个后台服务里根据分类名称查询分类编号，然后将“产品分类”这一列的值替换为分类编号，这样就可以保证导入的数据是有效的，计量单位也可以参考这种方式处理；

(2) 判断当前行数据是否有效，如果无效，可以通过响应对象 res 返回参数 ignore，并且值设置为 true，告诉导入程序，忽略该行数据，不执行导入；



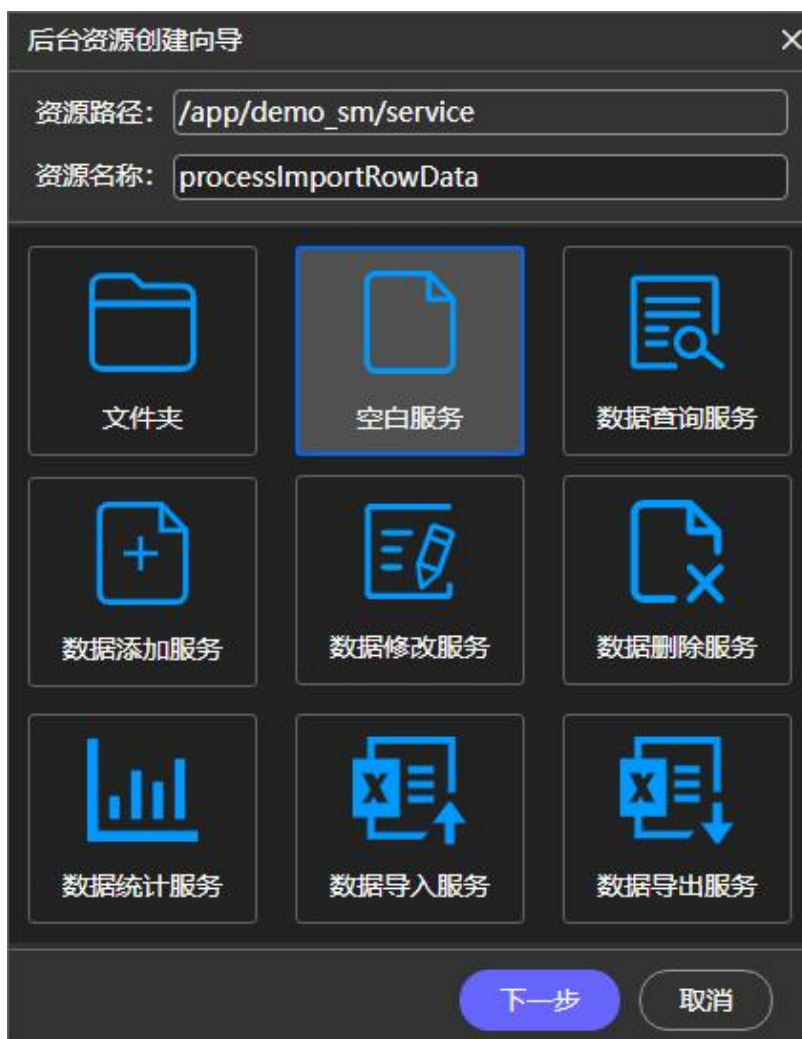
我们假设导入产品信息的 Excel 格式和内容如下：

|   | A    | B    | C    | D                 | E    | F      | G  |
|---|------|------|------|-------------------|------|--------|----|
| 1 | 产品编码 | 产品名称 | 产品分类 | 规格型号              | 计量单位 | 单价     | 备注 |
| 2 | 1002 | 拖把   | 日用品  | 大卫免手洗平板拖把         | 个    | 71.90  |    |
| 3 | 2001 | 方便面  | 食品   | 康师傅红烧牛肉面五连包129g*5 | 包    | 15.00  |    |
| 4 | 3001 | 电风扇  | 电器   | 美的落地空气循环电风扇       | 台    | 189.00 |    |

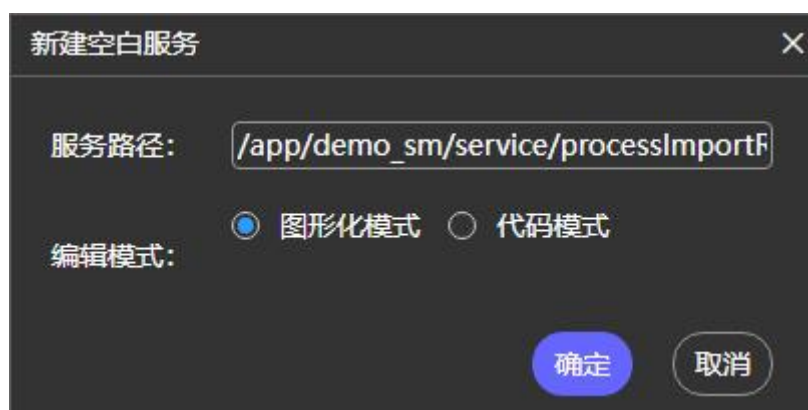
在继续执行下面的操作之前，请先在系统维护的数据字典内维护好相应的计量单位信息，并在产品信息列表的左侧里，添加好对应的产品分类信息，否则执行后续服务时查不到数据。

下面介绍具体如何实现在导入产品信息时关联查询产品分类编号和计量单位编码。

在 TaskBuilder 里，点击销售管理示例项目的后台服务里的 product 目录右侧的加号，打开如下图所示的“后台资源创建向导”，选中“空白服务”，资源名称设置为：processImportRowData。



然后点下一步，出现如下对话框：



在此不用做任何修改，直接点确定即可创建一个空白的后台服务，然后按下图所示设置：



上图语句的具体含义说明如下：

- (1) 先声明两个变量，并当前行的分类名称和计量单位名称存到这两个变量中：

```
typeName = req.rowData[2]
```

```
unitName = req.rowData[4]
```

上面语句里的 req.rowData 是导入程序传给该服务的当前行数据，类型为数组，里面的元素是当前行每一列的数据，例如：['1002','拖把','日用品','大卫免手洗平板拖把','个','71.90','']

(2) 再拖拽一个数据查询语句，用该行的产品分类名称查询分类编号，设置界面如下：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '基本信息' (Basic Information) tab selected. The '数据模型' (Data Model) is set to 'demo\_sm/model/demo\_sm\_product\_type.tdm'. The '翻页设置' (Pagination Settings) section has '启用翻页' (Enable Pagination) checked, '每页最多显示' (Max items per page) set to 20, and '条记录' (Records). The '查询数量' (Query Quantity) is set to '最多查询' (Max query) with a value of 1. The '重复设置' (Repeat Settings) has '去掉重复记录' (Remove duplicate records) checked. The '查询结果参数名称' (Query result parameter name) is set to 'rows1'. There are '确定' (Confirm) and '取消' (Cancel) buttons at the bottom right.

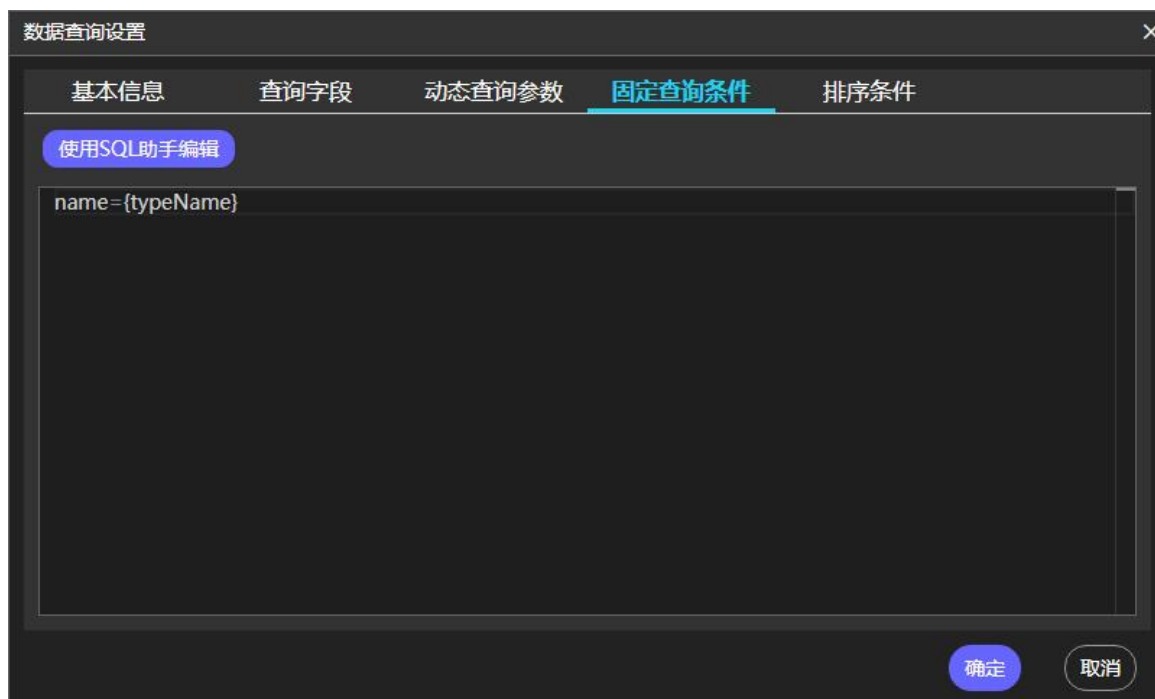
基本信息里,数据模型选择 demo\_sm\_product\_type.tdm,查询结果参数名称设置为:rows1。

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '查询字段' (Query Fields) tab selected. There are two buttons at the top: '添加自定义字段' (Add custom field) and '添加关联字段' (Add associated field). Below is a table with columns: '字段名称' (Field Name), '字段备注' (Field Remark), '字段格式(SQL, 双击可编辑)' (Field Format (SQL, double-click to edit)), '字段别名' (Field Alias), and '汇总' (Summary). The table contains the following rows:

| 字段名称                                   | 字段备注   | 字段格式(SQL, 双击可编辑) | 字段别名 | 汇总 |
|----------------------------------------|--------|------------------|------|----|
| <input checked="" type="checkbox"/> id | 分类编号   |                  |      | ▼  |
| <input type="checkbox"/> code          | 分类编码   |                  |      | ▼  |
| <input type="checkbox"/> name          | 分类名称   |                  |      | ▼  |
| <input type="checkbox"/> parent_id     | 上级分类编号 |                  |      | ▼  |
| <input type="checkbox"/> sort_num      | 分类序号   |                  |      | ▼  |
| <input type="checkbox"/> status        | 分类状态   |                  |      | ▼  |

There are '确定' (Confirm) and '取消' (Cancel) buttons at the bottom right.

在查询字段里选中 id 字段。



固定查询条件设置为：name={typeName}，表示查询名称等于变量 typeName 的数据。

(3) 然后在该查询语句里拖拽一个变量赋值语句，将当前行里的产品分类这一列的数据设置为 0，表示如果没有查询到对应的产品分类，就将分类编号设置为默认值 0；

(4) 然后再拖拽一个判断语句，判断上面查询语句的查询结果数组 rows1 的长度是否大于 0，如果大于 0，说明查到了对应的分类数据；

(5) 然后在判断语句里拖拽一个赋值语句，把当前行里的产品分类这一列的数据设置为查询到的分类编号：rows1[0].id；

(6) 然后再拖一个数据查询语句，根据这一列的计量单位名称查询单位编码，具体操作界面就不截图了，可以参考上面的查询产品分类的操作，不一样的地方包括：

- **数据模型：**选择 sys/model/sys\_data\_dic.tdm；
- **查询结果参数名称：**设置为 rows2，跟上面的查询结果参数名称不要一样；
- **查询字段：**选中 code；
- **固定查询条件：**设置为 name={unitName}；

该查询语句里的子语句也基本与上面查询产品分类信息里的一样，就不详细解释了。

这个后台服务里的语句都设置好后，回到产品信息列表页面的设计界面，选中产品列表上部面板里的导出组件，点击属性里的“数据导入设置”，打开如下所示对话框：

数据导入设置

提交数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务

**数据模型** 请求参数 修改条件

安全设置: ☐ 不校验是否登录 ☐ 不检查SQL注入 标题行位置: 在第  行

记录存在时: ☒ 修改现有值 ☐ 删除重建 ☐ 不做任何处理 导入数据为空时: ☐ 将现有数据也修改为空

数据模型路径:

对每行数据进行特殊处理的后台服务:

|                                     | 字段名称     | 数据类型 | Excel列标题 | 必填                                  | 主键                                  | 计算表达式       | 默认值 | 可选值(英文逗号分隔) |
|-------------------------------------|----------|------|----------|-------------------------------------|-------------------------------------|-------------|-----|-------------|
| <input checked="" type="checkbox"/> | code     | 文本   | 产品编码     | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |             |     |             |
| <input checked="" type="checkbox"/> | name     | 文本   | 产品名称     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |             |     |             |
| <input checked="" type="checkbox"/> | type_id  | 整数   | 产品分类     | <input type="checkbox"/>            | <input type="checkbox"/>            |             | 0   |             |
| <input checked="" type="checkbox"/> | sort_num | 整数   | 产品序号     | <input type="checkbox"/>            | <input type="checkbox"/>            | {_RowIndex} |     |             |
| <input checked="" type="checkbox"/> | model    | 文本   | 规格型号     | <input type="checkbox"/>            | <input type="checkbox"/>            |             |     |             |
| <input checked="" type="checkbox"/> | unit     | 文本   | 计量单位     | <input checked="" type="checkbox"/> | <input type="checkbox"/>            |             |     |             |
| <input checked="" type="checkbox"/> | price    | 数字   | 单价       | <input type="checkbox"/>            | <input type="checkbox"/>            |             | 0   |             |

在“对每行数据进行特殊处理的后台服务”输入框右侧点击选择按钮，选中前面创建的这个后台服务：demo\_sm/service/product/processImportRowData.tbs，然后点确定，然后保存页面，再导入数据，就能自动生成产品分类编号和计量单位的编码了。

#### 4.13.1.3 设置 Excel 导入组件的“完成导入后”事件

在数据导入完成后，要自动刷新产品信息列表，否则看不到新导入的数据，实现方法如下：

在列表页面中选中导入组件，然后双击事件设置中的“完成导入后”事件输入框，打开该事件的可视化编排界面，然后从语句库的组件操作中拖拽一个“调用组件方法”语句，将组件设置为 grid1，方法设置为：重新加载数据，然后保存页面，这样在导入完数据后就会自动刷



新产品列表，设置界面如下所示：



至此，整个产品信息导入的功能就开发完毕了，然后可以预览测试，具体操作就不详述了。

## 4.13.2 实现产品信息导出功能

接下来我们实现产品信息的导出功能，操作方法如下：

### 4.13.2.1 添加和设置 Excel 导出组件

在 TaskBuilder 里打开产品信息列表页面，从组件库的导入导出中拖拽“Excel 导出”组件到产品列表上面的面板组件中，在上面的样式设置栏的边距和边框里，将外边距里的左边距设置为 15，然后点属性里的“数据导出设置”，打开如下所示对话框：

| ■                                   | 字段名称      | 字段格式(双击可编辑)          | 字段别名      | Excel列标题 | 列宽  | 数据字典(双击选择) |
|-------------------------------------|-----------|----------------------|-----------|----------|-----|------------|
| <input checked="" type="checkbox"/> | code      |                      |           | 产品编码     | 80  |            |
| <input checked="" type="checkbox"/> | name      |                      |           | 产品名称     | 100 |            |
| <input checked="" type="checkbox"/> | type_name | (select name from de | type_name | 产品分类     | 80  |            |
| <input checked="" type="checkbox"/> | model     |                      |           | 规格型号     | 240 |            |
| <input checked="" type="checkbox"/> | unit      |                      |           | 计量单位     | 60  | unit       |
| <input checked="" type="checkbox"/> | price     |                      |           | 单价       | 80  |            |
| <input checked="" type="checkbox"/> | comment   |                      |           | 备注       | 200 |            |

在此，需要设置以下参数：

- **应用代码：**导出的 Excel 文件的信息会以附件形式保存到系统附件表中，以备将来查证，



为了区分各个附件所属的功能模块，要求每条附件信息必须设置一个应用代码，具体内容只要能区分是哪个应用的附件即可，但只能是字符串、数字和下划线，最多 20 个字符。

- **导出后的文件名：**用来设置导出的 Excel 的文件名，里面可以采用 **{日期格式}** 的形式插入当前的日期时间值。
- **数据模型：**选择产品信息的数据模型 demo\_sm\_product.tdm；

然后按照下表所示选择和设置要导出的字段：

| 字段名称    | Excel 列标题 | 列宽  | 数据字典 |
|---------|-----------|-----|------|
| code    | 产品编码      | 80  |      |
| name    | 产品名称      | 100 |      |
| model   | 规格型号      | 240 |      |
| unit    | 计量单位      | 60  | unit |
| price   | 单价        | 80  |      |
| comment | 备注        | 200 |      |

然后再添加一个关联字段，关联查询 demo\_sm\_product\_type 里的 name 字段：

**添加关联字段**

关联对象类型：  
☒ 表 ☐ 视图

关联数据对象：  
 demo sm product typ ▼

关联到主表字段：  
 name ▼

关联字段别名：  
 type\_name

关联字段备注：  
 产品分类

关联表字段：  
 双击插入到关联条件中  
 id  
 code  
 name  
 parent\_id  
 sort\_num  
 status

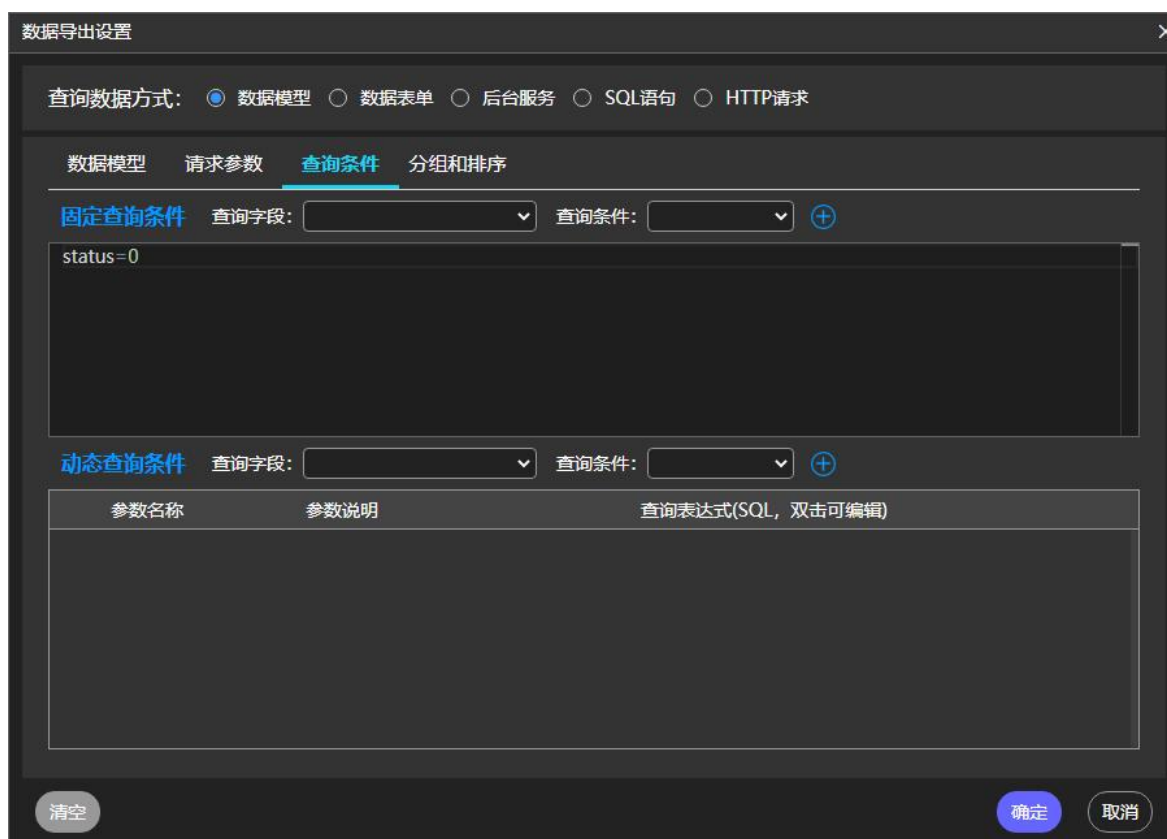
主表字段：  
 双击插入到关联条件中  
 id  
 code  
 name  
 type\_id  
 sort\_num  
 model  
 unit  
 price  
 thumbnail  
 comment

关联条件：  
 and or = <> > < >= <= like in 为空 不为空  
 id = demo\_sm\_product.type\_id

确定 取消

关联条件为：`id=demo_sm_product.type_id`，并将字段别名设置为 `type_name`，Excel 列标题设置为产品分类，宽度设置为 80。

然后点击“查询条件”选项卡，在里面将固定查询条件设置为：`status=0`，表示只查状态正常的产品信息：



数据导出设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 **查询条件** 分组和排序

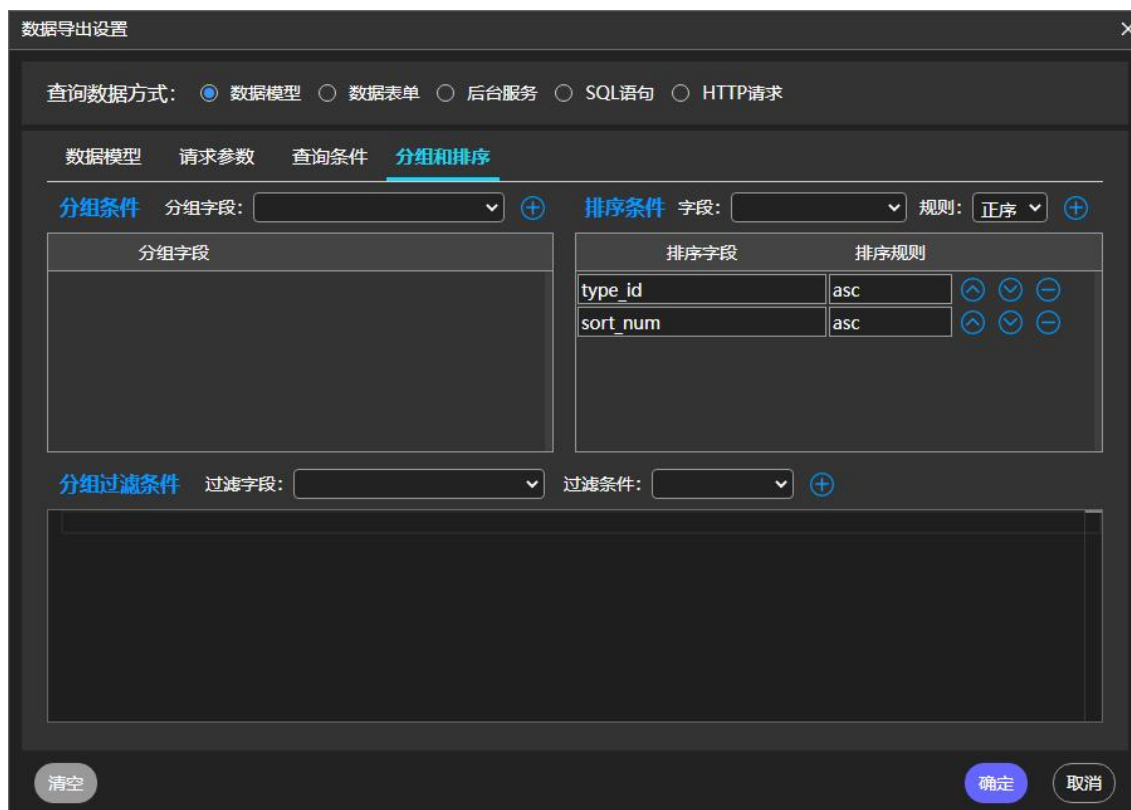
**固定查询条件** 查询字段:  查询条件:

status=0

**动态查询条件** 查询字段:  查询条件:

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

然后点击“分组和排序”选项卡，分别将产品分类编号和产品序号添加为排序条件，且都设置为按正序排列：

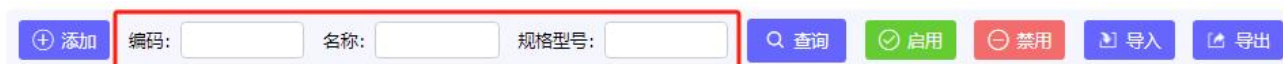


都设置完成后，点击确定按钮，保存页面，即可预览测试导出功能。

#### 4.13.2.2 根据用户输入的查询条件导出数据

上面设置的导出功能会导出全部状态正常的产品信息，如果想根据用户在页面里输入的查询条件导出满足这些条件的数据，则可以按以下方法设置：

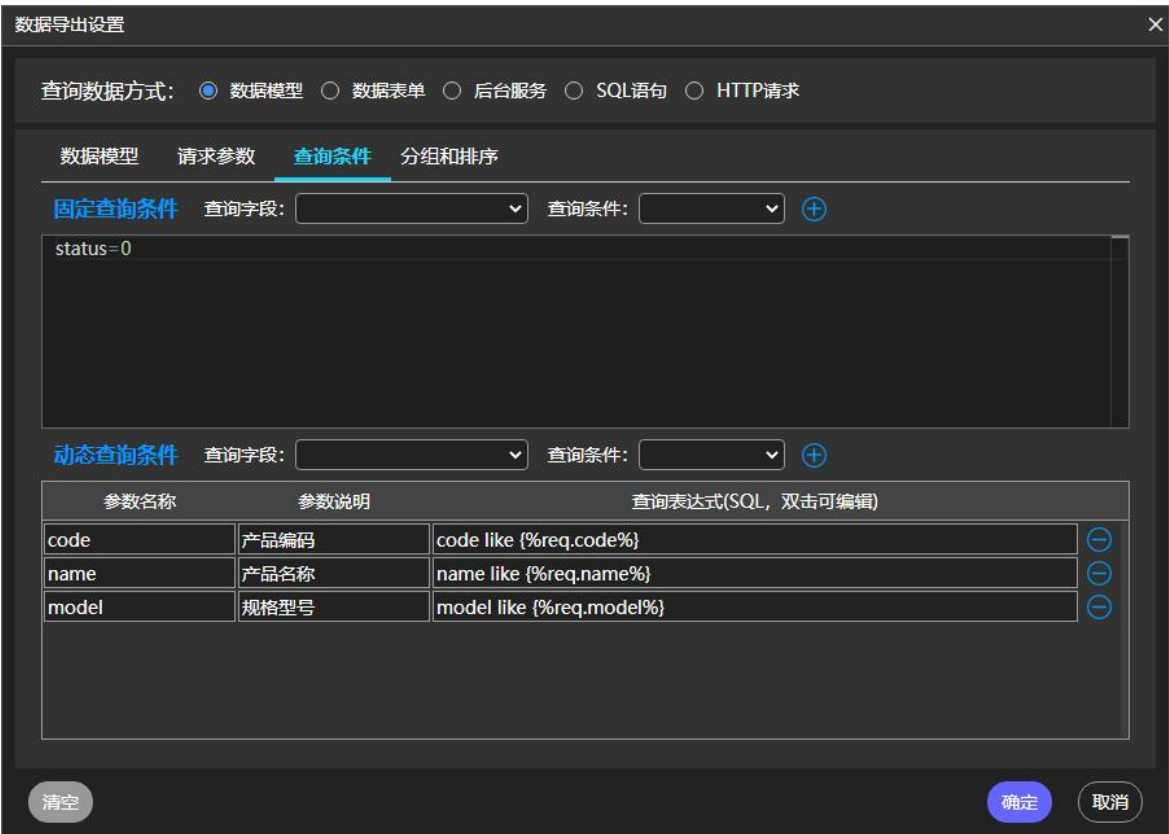
我们前面创建的产品列表页面里，在产品列表上部提供了产品编码、产品名称和规格型号三个查询输入项，如下图所示：



现在我们选中导出组件，然后在属性设置里，点击“数据导出设置”，然后在弹出的对话框里点击“请求参数”选项卡，将上面三个查询输入项的组件编号作为请求参数添加到参数列表里，并将参数类型设置为组件值，如下图所示：



然后再点击“查询条件”选项卡，在动态查询条件里，添加这三个查询输入项对应的查询条件，都设置为模糊查询，如下图所示：



然后点确定，这样就将页面里的查询输入项与导出组件建立了连接，再点导出时，如果用户在查询输入项里输入了相关查询条件，就会根据这些条件导出相应的数据。

## 4. 14 发布产品管理应用

整个产品管理模块开发完成后，就可以在后台管理系统的系统维护的应用管理中，添加对应的应用信息，相关信息设置如下：

- **应用代码：** 设置为 demo\_sm\_product；
- **应用名称：** 设置为产品管理；
- **应用分类：** 选前面创建的“示例项目”这个类；
- **支持终端：** 选择 Web 端；
- **web 端首页 URL 路径：** /app/demo\_sm/product/index.tfp；

添加完应用后，还可以添加该应用的操作信息和访问权限，具体如何设置请参考前面客户管理模块里的相关介绍，这里就不再复述。

## 5、销售订单模块开发

### 5.1 本节知识点

- (1) 编码规则组件的使用；
- (2) 日期选择组件的使用；
- (3) 弹窗选择数据的实现；
- (4) 可编辑表格的使用；
- (5) 输入项自动计算表达式的设置；
- (6) 主明细表数据的提交；

(7) 多列表页面数据懒加载的实现;

## 5.2 创建数据模型

销售订单需要创建销售主表和明细表两个数据模型，主表用来存储销售订单的基本信息，包括订单编码、经办人信息、客户信息、金额合计等，明细表用来存储销售的具体产品信息，包括产品编码、产品名称、规格型号、单价、数量、计量单位和金额等，具体见下面的介绍。

### 5.2.1 创建销售订单主表数据模型

请使用 TaskBuilder 的新建数据模型向导在 demo\_sm 示例项目的数据模型里，创建销售订单主表对应的数据模型：demo\_sm\_sale\_order.tdm，然后按照下表所示创建和设置字段，具体操作就不再详细讲解，请参考客户管理模块里的相关介绍。

| 字段名             | 数据类型 | 长度  | 字段备注  | 非空 | 唯一 | 说明     |
|-----------------|------|-----|-------|----|----|--------|
| id              | 整数   | 11  | 订单编号  | 是  | 是  | 主键，自增长 |
| code            | 字符串  | 10  | 订单编码  | 是  |    |        |
| operator_id     | 整数   | 11  | 经办人编号 | 是  |    |        |
| order_date      | 日期   |     | 下单日期  | 是  |    |        |
| customer_id     | 整数   | 11  | 客户编号  | 是  |    |        |
| contacts        | 字符串  | 10  | 联系人   | 是  |    |        |
| contact_number  | 字符串  | 20  | 联系电话  | 是  |    |        |
| order_amount    | 金额   | 16  | 订单金额  | 是  |    |        |
| discount_amount | 金额   | 16  | 优惠金额  |    |    |        |
| actual_amount   | 金额   | 16  | 实收金额  | 是  |    |        |
| creator_id      | 整数   | 11  | 创建人编号 | 是  |    |        |
| create_time     | 日期时间 |     | 创建时间  | 是  |    |        |
| auditor_id      | 整数   | 11  | 审核人编号 |    |    |        |
| audit_time      | 日期时间 |     | 审核时间  |    |    |        |
| audit_opinion   | 字符串  | 200 | 审核意见  |    |    |        |
| deletor_id      | 整数   | 11  | 删除人编号 |    |    |        |



|             |      |     |      |   |  |                                               |
|-------------|------|-----|------|---|--|-----------------------------------------------|
| delete_time | 日期时间 |     | 删除时间 |   |  |                                               |
| comment     | 字符串  | 500 | 备注   |   |  |                                               |
| status      | 微整数  | 4   | 分类状态 | 是 |  | 0 表示待审核, 1 表示审核通过, 2 表示审核未通过, -1 表示已删除, 默认为 0 |

销售订单主表数据模型创建完的界面如下图所示:

| <input type="checkbox"/> | 字段名             | 字段类型 | 长度  | 精度 | 主键                                  | 非空                                  | 自增                                  | 唯一                       | 默认值 | 备注    |
|--------------------------|-----------------|------|-----|----|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-----|-------|
| <input type="checkbox"/> | id              | 整数   | 11  |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |     | 订单编号  |
| <input type="checkbox"/> | code            | 字符串  | 20  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 订单编码  |
| <input type="checkbox"/> | operator_id     | 整数   | 11  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 经办人编号 |
| <input type="checkbox"/> | order_date      | 日期   |     |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 下单日期  |
| <input type="checkbox"/> | customer_id     | 整数   | 11  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 客户编号  |
| <input type="checkbox"/> | contacts        | 字符串  | 10  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 联系人   |
| <input type="checkbox"/> | contact_number  | 字符串  | 20  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 联系电话  |
| <input type="checkbox"/> | order_amount    | 金额   | 16  | 2  | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 订单金额  |
| <input type="checkbox"/> | discount_amount | 金额   | 16  | 2  | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 优惠金额  |
| <input type="checkbox"/> | actual_amount   | 金额   | 16  | 2  | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 实收金额  |
| <input type="checkbox"/> | creator_id      | 整数   | 11  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 创建人编号 |
| <input type="checkbox"/> | create_time     | 日期时间 |     |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 创建时间  |
| <input type="checkbox"/> | auditor_id      | 整数   | 11  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 审核人编号 |
| <input type="checkbox"/> | audit_time      | 日期时间 |     |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 审核时间  |
| <input type="checkbox"/> | audit_opinion   | 字符串  | 200 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 审核意见  |
| <input type="checkbox"/> | deletor_id      | 整数   | 11  |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 删除人编号 |
| <input type="checkbox"/> | delete_time     | 日期时间 |     |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 删除时间  |
| <input type="checkbox"/> | status          | 微整数  | 4   |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 状态    |
| <input type="checkbox"/> | comment         | 字符串  | 200 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 备注    |

## 5.2.2 创建销售订单明细表数据模型

继续创建销售订单明细表对应的数据模型: demo\_sm\_sale\_order\_detail.tdm, 然后按照下表所示创建和设置字段:

| 字段名      | 数据类型 | 长度 | 字段备注 | 非空 | 唯一 | 说明      |
|----------|------|----|------|----|----|---------|
| id       | 整数   | 11 | 明细编号 | 是  | 是  | 主键, 自增长 |
| order_id | 整数   | 11 | 订单编号 | 是  |    |         |

|            |     |     |      |   |  |                          |
|------------|-----|-----|------|---|--|--------------------------|
| product_id | 整数  | 11  | 产品编号 | 是 |  |                          |
| price      | 金额  | 16  | 单价   | 是 |  |                          |
| quantity   | 整数  | 11  | 数量   | 是 |  | 在实际业务中，数量可能是小数，这里为了简化用整数 |
| unit       | 字符串 | 2   | 计量单位 | 是 |  |                          |
| amount     | 金额  | 16  | 金额   | 是 |  |                          |
| comment    | 字符串 | 200 | 备注   |   |  |                          |

创建后的数据模型界面如下图所示：

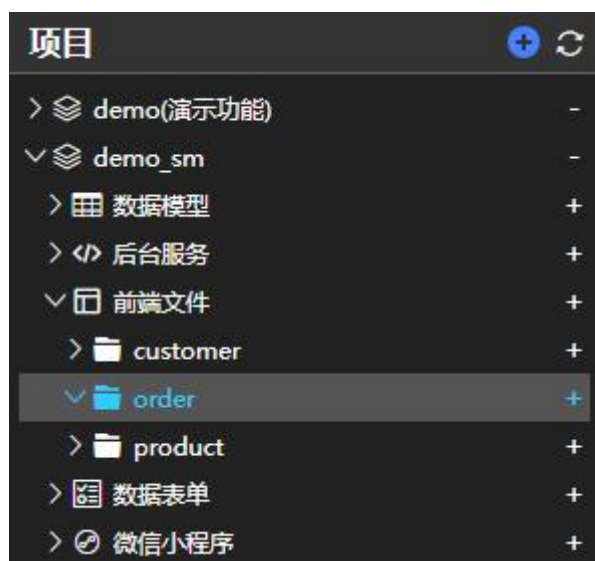
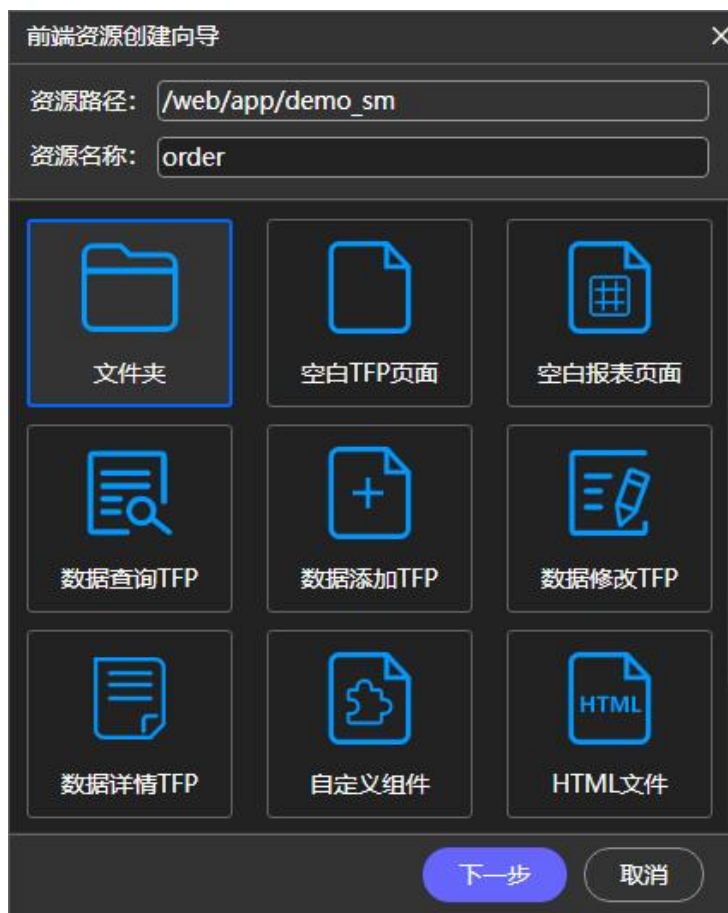
| <input type="checkbox"/> | 字段名        | 字段类型 | 长度 | 精度 | 主键                                  | 非空                                  | 自增                                  | 唯一                       | 默认值 | 备注   |
|--------------------------|------------|------|----|----|-------------------------------------|-------------------------------------|-------------------------------------|--------------------------|-----|------|
| <input type="checkbox"/> | id         | 整数   | 11 |    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> |     | 明细编号 |
| <input type="checkbox"/> | order_id   | 整数   | 11 |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 订单编号 |
| <input type="checkbox"/> | product_id | 整数   | 11 |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 产品编号 |
| <input type="checkbox"/> | price      | 金额   | 16 | 2  | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 单价   |
| <input type="checkbox"/> | quantity   | 整数   | 11 |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 数量   |
| <input type="checkbox"/> | unit       | 字符串  | 2  |    | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 计量单位 |
| <input type="checkbox"/> | amount     | 金额   | 16 | 2  | <input type="checkbox"/>            | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input type="checkbox"/> |     | 金额   |
| <input type="checkbox"/> | comment    | 字符串  | 45 |    | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/> |     | 备注   |

## 5.3 创建销售管理文件夹

点击销售管理示例项目“前端文件”右侧的加号按钮：

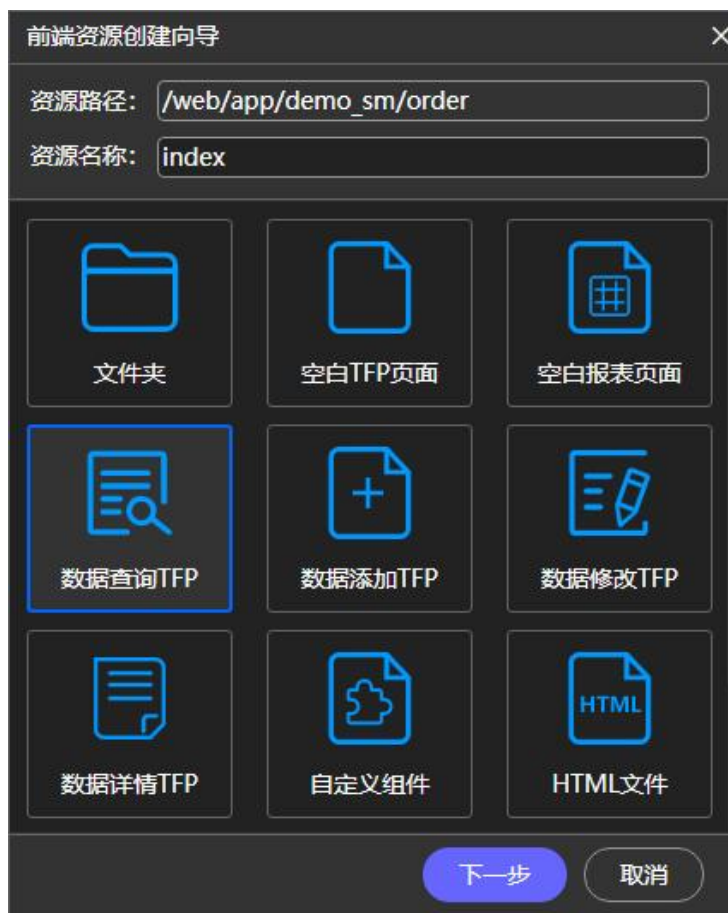


打开如下图所示的“前端资源创建向导”对话框，在该对话框中，选中“文件夹”，在资源名称输入框中输入：order，然后点“下一步”按钮，就可以在销售管理示例项目的前端创建一个名为“order”的文件夹，以便存放所有订单管理有关的前端文件。



## 5.4 创建销售订单列表页面

在销售管理示例项目的前端文件里，点击上面创建的 order 文件夹右侧的加号按钮，打开“前端资源创建向导”，资源类型选择“数据查询 TFP”：



然后点下一步，打开如下所示的“创建数据查询 TFP”对话框：



在此，先选择销售订单主表数据模型：demo\_sm\_sale\_order.tdm，然后设置页面标题：销

售订单管理，其他参数默认，然后点击下一步按钮，打开下图所示的窗口：



在此选择以下字段：

id、code、order\_date、contacts、contact\_number、order\_amount、discount\_amount、actual\_amount、create\_time、audit\_time、delete\_time、status、comment

然后添加以下字段的关联字段：

| 关联表              | 关联字段 | 字段别名          | 字段备注 | 关联条件                                |
|------------------|------|---------------|------|-------------------------------------|
| demo_sm_customer | name | customer_name | 客户名称 | id = demo_sm_sale_order.customer_id |
| sys_user         | name | operator_name | 经办人  | id = demo_sm_sale_order.operator_id |
| sys_user         | name | creator_name  | 创建人  | id = demo_sm_sale_order.creator_id  |
| sys_user         | name | auditor_name  | 审核人  | id = demo_sm_sale_order.auditor_id  |
| sys_user         | name | deletor_name  | 删除人  | id = demo_sm_sale_order.deletor_id  |

然后再点下一步，按照下表设置要显示的字段：

| 列名   | 列类型 | 数据绑定格式          | 文本显示格式        | 对齐方式 | 宽度  |
|------|-----|-----------------|---------------|------|-----|
| 订单编码 | 链接  | {code}          |               |      | 100 |
| 经办人  | 文本  | {operator_name} |               |      | 60  |
| 下单日期 | 文本  | {order_date}    | 日期：yyyy-MM-dd |      | 100 |
| 客户名称 | 文本  | {customer_name} |               |      | 200 |



|      |    |                   |                |     |     |
|------|----|-------------------|----------------|-----|-----|
| 联系人  | 文本 | {contacts}        |                |     | 60  |
| 联系电话 | 文本 | {contact_number}  |                |     | 120 |
| 订单金额 | 文本 | {order_amount}    | 金额             | 右对齐 | 80  |
| 优惠金额 | 文本 | {discount_amount} | 金额             | 右对齐 | 80  |
| 实收金额 | 文本 | {actual_amount}   | 金额             | 右对齐 | 80  |
| 创建人  | 文本 | {creator_name}    |                |     | 60  |
| 创建时间 | 文本 | {create_time}     | 日期: yyyy-MM-dd |     | 100 |
| 备注   | 文本 | {comment}         |                | 左对齐 | 200 |

最终设置完的界面如下图：



然后再点下一步，在此设置以下动态查询参数：

| 参数名         | 参数说明 | 输入项类型 | 查询表达式                         |
|-------------|------|-------|-------------------------------|
| code        | 订单编码 | 单行文本  | code like {%req.code%}        |
| operator_id | 经办人  | 用户    | operator_id={req.operator_id} |

设置完后的界面如下图所示：



创建数据查询TFP

第四步：设置动态查询参数

查询字段：

查询条件：

+

| 参数名称        | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)                   | 默认值 | 数据字典(双击选择) | 可选值          |
|-------------|------|-------|----|-------------------------------|-----|------------|--------------|
| code        | 订单编码 | 单行文本  |    | code like {%req.code%}        |     |            | <div>⊖</div> |
| operator_id | 经办人  | 用户    |    | operator_id={req.operator_id} |     |            | <div>⊖</div> |

上一步

下一步

取消

然后再点下一步，会出现“设置固定查询条件”界面，在此不需要设置任何条件，直接点下一步，然后出现“设置排序条件”界面，在此添加按订单编码倒序排列的排序规则：

创建数据查询TFP

第六步：设置排序条件

排序字段：

排序规则：

正序

倒序

+

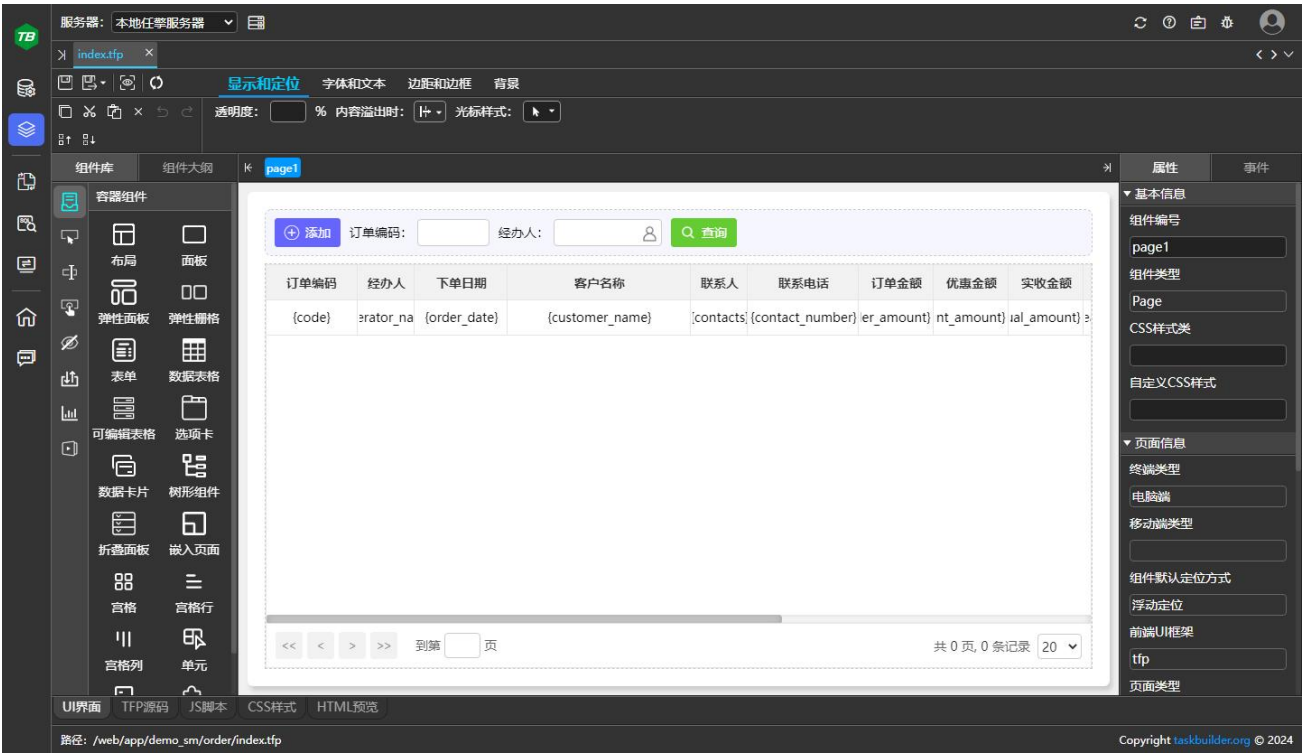
| 排序字段 | 排序规则                                                 |
|------|------------------------------------------------------|
| code | desc <div><div>⬆</div><div>⬇</div><div>⬇</div></div> |

上一步

确定

取消

然后点确定按钮，即可完成订单信息列表页面的创建，界面如下图所示：



## 5.5 设置销售订单编码规则

### 5.5.1 编码规则简介

我们在创建各种业务单据时，一般会为每张单据设置一个唯一的单据编码，这个单据编码可以包含业务类型、日期时间、部门和流水号等丰富信息，相对于自动增长的纯数字型编号，会更加方便理解，让用户看到单据编码，就能知道是什么业务、什么时间的创建的。

为了方便开发者实现自动生成单据编码的功能，在任擎后台管理系统的系统维护内，提供了一个专门的“编码规则”模块，可以在此为各个业务单据配置好相应的编码规则，然后在设置单据添加页面时，就可以使用组件库里的编码规则组件，生成这个单据的编码，下面介绍具体如何实现。

### 5.5.2 添加销售订单编码规则

登录任擎后台管理系统，点击系统维护内的编码规则模块，即可打开编码规则列表页面，

然后点击该页面中的添加按钮，打开“添加编码规则”抽屉页，如下图所示：

在该抽屉页中，可以设置编码规则的基本信息和编码格式，基本信息包括以下参数：

- **规则代码：**用来给当前编码规则设置一个字符串，用来与其他编码规则区分，不能重复。
- **规则名称：**用来设置当前编码规则的名称；
- **流水号长度：**用来设置当前编码规则的流水号位数，可以选 2、4、6、8、10 这 5 种长度，默认为 4 位，开发者可以根据数据量大小自己决定。如果编码格式里包含了年月日这种时间数据，且设置了按年月日清零，则流水号的位数只需要考虑在一个清零周期内的最大值即可，例如编码格式里包含了年和月的信息，且设置了按月清零，如果这类单据每月最多就几千张，则流水号长度设置为 4 位即可，如果每月有上万张，则设置为 6 位即可。
- **清零方式：**用来设置流水号清零重新自增的方式，可以选按年、月和日三种时间周期清零。
- **更新方式：**用来设置单据编码的更新方式，可以选读取时更新和写入时更新，如果选读取时更新，则在打开单据填报页面，调用获取编码的后台服务器时，就会自动更新流水号，生成最新的单据编码并显示在单据填报页面里，然后在保存单据时，就会将这个编码保存到单据表中，这种方式是默认的方式，也是最简单的方式，但这种方式存在两个问题：一

是用户打开单据填报界面后，如果没有保存单据，点取消或关闭了，则该单据编码就浪费了，单据表里就会出现不连续的单据编码，另外一个问题是单据编码和保存时间的顺序可能不一致，例如 A 用户先打开单据填报界面，他还没来得及保存，B 用户也打开了这个单据的填报界面，而且很快就填写完毕，并保存了，然后 A 用户才提交，这时，数据库的订单表中，如果看单据编码里的流水号，A 用户提交的流水号比 B 用户的小，但看实际保存时间，A 用户提交的又比 B 用户提交的晚。如果选择写入时更新，则打开单据填报页面时，调用获取编码信息的后台服务获得的单据编码是一个临时性的编码，并且不会更新该单据的流水号，需要在提交单据数据时，再调用一次获取编码信息的后台服务，并将参数 `updateType` 设置为 R，这时才会真正的生成新的单据编码，然后保存到单据表中，这样就能避免出现前面说的那两个问题，但这种方式使用起来要麻烦一些，需要请求两次获取单据编码的服务，所以默认没有采用该方式。

- **流水号类型：**用来设置流水号按哪种级别的组织结构生成，可以选按组织生成、按部门生成和按用户生成，默认为按组织生成，即对于这个编码规则来说，整个组织使用一套流水号，如果选按部门生成，则每个部门都有一套从 1 开始自动增长的流水号，如果选按用户生成，则每个用户都有一套流水号。

编码规则的基本信息设置好后，还需要设置编码格式，即编码具体由哪些内容构成，点击编码格式列表上面的添加按钮，可以添加多行格式组成一个编码规则，在每一行内，需要先选择具体的格式，有以下几种格式可以选择：

- **字符串：**即固定的一个或多个字符，一般采用业务功能的英文或拼音简称的大写字母，以便通过该字符串就能判断单据的类型，例如销售单：XS、采购单：CG、出库单：CK 等，需要将这些字符输入到后面的格式内容输入框内。其他格式不需要手动输入格式内容，选择完格式类型，会自动生成格式内容。

- **流水号**：即按照上面基本信息里设置的相关参数自动生成的流水号。
- **日期信息**：即指定格式的日期信息，可选 4 位年份、2 位年份、2 位月份、2 位日、年月日，这里的日期是指生成编码时服务器的当前日期。
- **部门信息**：指当前用户所在部门的信息，包括部门代码和部门名称；
- **用户信息**：指当前用户的信息，包括用户代码和用户名称。

以示例项目的销售订单为例，我们将该单据编码规则的基本信息设置为：

- **规则代码**：demo\_sm\_sale\_order\_code
- **规则名称**：示例项目销售订单编码
- **流水号长度**：4
- **清零方式**：按月清零
- **更新方式**：读取时更新
- **流水号类型**：按组织生成

编码格式设置成以下几个格式的组合：

- (1) **字符串**：XS
- (2) **4 位年份**：{YYYY}
- (3) **2 位月份**：{MM}
- (4) **流水号**：{XXXX}

最终设置完的界面如下图所示：

### 基本信息

规则代码：\*  
demo\_sm\_sale\_order\_code

规则名称：\*  
示例项目销售订单编码

流水号长度：\*  
☐ 2 ☒ 4 ☐ 6 ☐ 8 ☐ 10

清零方式：\*  
按月清零

更新方式：\*  
读取时更新

流水号类型：\*  
按组织生成

### 编码格式

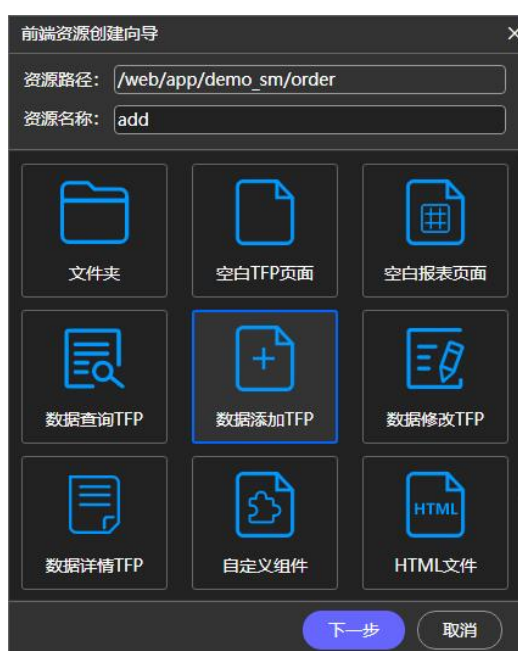
添加

| 格式类型 | 格式内容   |       |
|------|--------|-------|
| 字符串  | XS     | ⬆ ⬆ ⬆ |
| 4位年份 | {YYYY} | ⬆ ⬆ ⬆ |
| 2位月份 | {MM}   | ⬆ ⬆ ⬆ |
| 流水号  | {XXXX} | ⬆ ⬆ ⬆ |

然后点右上角的确定按钮，即可完成该编码规则的创建。

## 5.6 创建添加销售订单页面

点击销售管理示例项目的前端文件 order 目录右侧的加号，打开如下图所示的“前端资源创建向导”：





在上面的对话框中选择“数据添加 TFP”，然后点下一步，打开如下所示窗口：

创建数据添加TFP

保存路径: /web/app/demo\_sm/order/add.tfp

数据模型: demo\_sm/model/demo\_sm\_sale\_order.tdm

选择...

页面标题: 添加销售单

页面类型: 抽屉页

窗口大小: 宽

高

终端类型: 电脑端

背景颜色: 浅色

定位方式: 浮动定位

输入项文本标签显示位置: ☒ 输入项上部 ☐ 输入项左侧

每行显示输入项数量: 3

仅支持PC

数据添加服务: ☒ 自动创建 ☐ 使用已有服务

选择...

前端页面设置字段

后台服务设置字段

| <input checked="" type="checkbox"/> | 字段              | 输入项名称 | 输入项类型 | 列数 | 必填                                  | 可选值 | 数据字典(双击选择) | 默认值      |
|-------------------------------------|-----------------|-------|-------|----|-------------------------------------|-----|------------|----------|
| <input checked="" type="checkbox"/> | code            | 订单编码  | 编码规则  |    | <input checked="" type="checkbox"/> |     |            | demo_sm_ |
| <input checked="" type="checkbox"/> | operator_id     | 经办人   | 用户选择  |    | <input checked="" type="checkbox"/> |     |            |          |
| <input checked="" type="checkbox"/> | order_date      | 下单日期  | 日期    |    | <input checked="" type="checkbox"/> |     |            |          |
| <input checked="" type="checkbox"/> | customer_id     | 所属客户  | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            |          |
| <input checked="" type="checkbox"/> | contacts        | 联系人   | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            |          |
| <input checked="" type="checkbox"/> | contact_number  | 联系电话  | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            |          |
| <input checked="" type="checkbox"/> | order_amount    | 订单金额  | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            | 0.00     |
| <input checked="" type="checkbox"/> | discount_amount | 优惠金额  | 单行输入框 |    | <input type="checkbox"/>            |     |            | 0.00     |
| <input checked="" type="checkbox"/> | actual_amount   | 实收金额  | 单行输入框 |    | <input checked="" type="checkbox"/> |     |            | 0.00     |
| <input checked="" type="checkbox"/> | comment         | 备注    | 多行输入框 | 3  | <input type="checkbox"/>            |     |            |          |

确定

取消

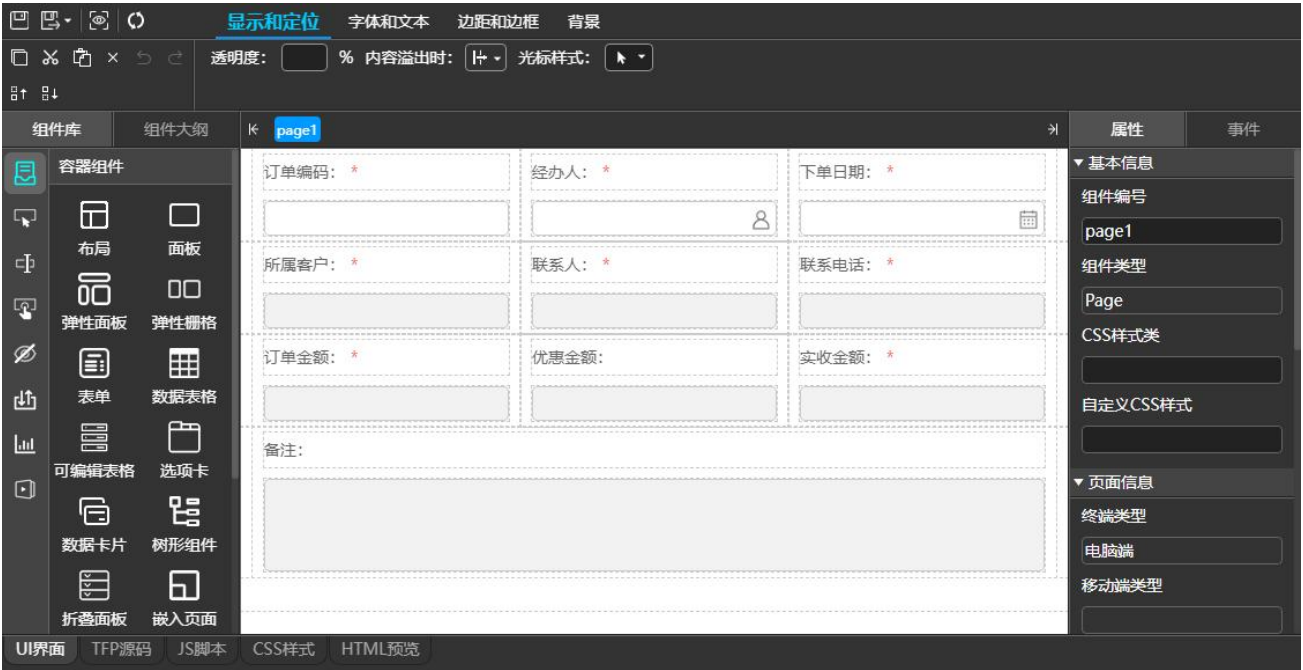
在此先选择订单主表数据模型：demo\_sm\_sale\_order.tdm，然后设置页面标题：添加订单，然后再按照下表的内容选择和设置“前端页面设置字段”：

| 字段名             | 输入项名称 | 输入项类型 | 列数 | 必填 | 默认值                     |
|-----------------|-------|-------|----|----|-------------------------|
| code            | 订单编码  | 编码规则  |    | 是  | demo_sm_sale_order_code |
| operator_id     | 经办人   | 用户选择  |    | 是  |                         |
| order_date      | 下单日期  | 日期    |    | 是  |                         |
| customer_id     | 所属客户  | 单行输入框 |    | 是  |                         |
| contacts        | 联系人   | 单行输入框 |    | 是  |                         |
| contact_number  | 联系电话  | 单行输入框 |    | 是  |                         |
| order_amount    | 订单金额  | 单行输入框 |    | 是  | 0.00                    |
| discount_amount | 优惠金额  | 单行输入框 |    |    | 0.00                    |
| actual_amount   | 实收金额  | 单行输入框 |    | 是  | 0.00                    |
| comment         | 备注    | 多行输入框 | 3  |    |                         |

然后点击“后台服务设置字段”，按下图所示设置：



然后点确定，即可完成“添加销售订单”页面的创建，如下图所示：



## 5.7 完善添加销售订单页面

上面用向导创建的订单添加页面还有很多不完善的地方，包括：

- (1) 内容比较多，没有分栏显示；
- (2) 客户信息需要手动录入，不能选择现有的客户；
- (3) 不能录入销售的商品明细；

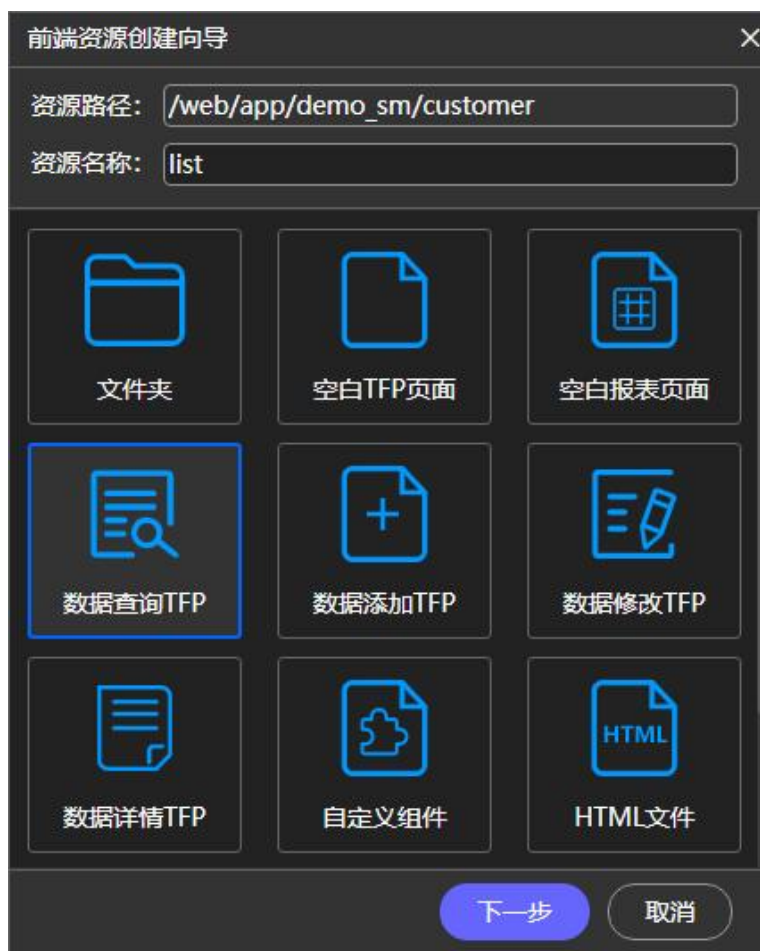
下面，我们来逐个完善：

## 5.7.1 实现弹窗选择客户

需要分以下两个大的步骤来实现：

### 5.7.1.1 创建选择客户对话框

在销售管理示例项目的前端文件中，点击 customer 文件夹右侧的加号按钮，打开下图所示“前端资源创建向导”：



在下面的资源类型列表选中“数据查询 TFP”，资源名称设置为：list，然后点下一步，打开如下图所示的“创建数据查询 TFP”对话框：

The dialog box is titled "创建数据查询TFP" (Create Data Query TFP). It has a close button (X) in the top right corner. The main content area is titled "第一步：设置基本信息" (Step 1: Set Basic Information). It contains several input fields and checkboxes:

- 保存路径: /web/app/demo\_sm/customer/list.tfp
- 数据模型: demo\_sm/model/demo\_sm\_customer.tdm (with a "选择..." button)
- 页面标题: 客户列表
- 页面类型: 对话框 (dropdown)
- 窗口大小: 宽 720, 高 480
- 终端类型: 电脑端 (dropdown)
- 背景颜色: 浅色 (dropdown)
- 定位方式: 浮动定位 (dropdown)
- 查询设置: ☒ 启用翻页, 每页最多显示 20 (dropdown), 条记录, 最多查询 ☐ 条记录, ☐ 去掉重复项
- 其他设置: ☐ 显示添加按钮, 添加页面文件名: add.tfp, 添加页面类型: 抽屉页 (dropdown), ☐ 显示修改按钮, 修改页面文件名: update.tfp, 修改页面类型: 抽屉页 (dropdown), ☐ 显示删除按钮, ☐ 查询条件显示在页面左侧 (仅支持PC端)
- 数据查询服务: ☒ 自动创建, ☐ 使用已有服务, 响应参数名: data (with a "选择..." button)
- 数据删除服务: ☒ 自动创建, ☐ 使用已有服务 (with a "选择..." button)

At the bottom right, there are two buttons: "下一步" (Next Step) and "取消" (Cancel).

在此，先选择数据模型：demo\_sm\_customer.tdm，页面标题设置为：客户列表，页面类型选择：对话框，宽度设置为：720，高度设置为：480，取消选中显示添加按钮、显示修改按钮、显示删除按钮这三个选项，然后点下一步，显示界面：

The dialog box is titled "创建数据查询TFP" (Create Data Query TFP). It has a close button (X) in the top right corner. The main content area is titled "第二步：设置要查询的字段" (Step 2: Set Fields to Query). It contains two buttons: "添加自定义字段" (Add Custom Field) and "添加关联字段" (Add Associated Field). Below these buttons is a table with the following columns: "字段名称" (Field Name), "字段备注" (Field Remark), "字段格式(双击编辑)" (Field Format (Double-click to edit)), "关联数据字典(双击选择)" (Associated Data Dictionary (Double-click to select)), and "字段别名" (Field Alias). The table contains 14 rows of data, with the first 10 rows having checkboxes in the first column. The last 4 rows have checkboxes in the first column.

|                                     | 字段名称        | 字段备注  | 字段格式(双击编辑)  | 关联数据字典(双击选择) | 字段别名 |
|-------------------------------------|-------------|-------|-------------|--------------|------|
| <input checked="" type="checkbox"/> | id          | 编号    | id          |              |      |
| <input checked="" type="checkbox"/> | code        | 编码    | code        |              |      |
| <input checked="" type="checkbox"/> | name        | 名称    | name        |              |      |
| <input type="checkbox"/>            | creator_id  | 创建人编号 | creator_id  |              |      |
| <input type="checkbox"/>            | create_time | 创建时间  | create_time |              |      |
| <input type="checkbox"/>            | status      | 状态    | status      |              |      |
| <input type="checkbox"/>            | salesman_id | 销售员编号 | salesman_id |              |      |
| <input checked="" type="checkbox"/> | contacts    | 联系人   | contacts    |              |      |
| <input checked="" type="checkbox"/> | telephone   | 联系电话  | telephone   |              |      |
| <input type="checkbox"/>            | scale       | 客户规模  | scale       |              |      |
| <input type="checkbox"/>            | grade       | 客户等级  | grade       |              |      |
| <input type="checkbox"/>            | source      | 客户来源  | source      |              |      |

At the bottom right, there are three buttons: "上一步" (Previous Step), "下一步" (Next Step), and "取消" (Cancel).

在此选中 id、code、name、contacts 和 telephone 这几个字段，然后点下一步，显示界面：

创建数据查询TFP

第三步：设置显示格式

| 列名   | 列类型 | 数据绑定格式(js表达式) | 文本显示格式 | 对齐方式 | 宽度  | 文字颜色 | 加粗 | 求和 |
|------|-----|---------------|--------|------|-----|------|----|----|
| 客户编码 | 文本  | {code}        |        | 居中对齐 | 120 |      |    |    |
| 客户名称 | 文本  | {name}        |        | 左对齐  | 300 |      |    |    |
| 联系人  | 文本  | {contacts}    |        | 居中对齐 | 80  |      |    |    |
| 联系电话 | 文本  | {telephone}   |        | 居中对齐 | 150 |      |    |    |

上一步

下一步

取消

在此，请按上图所示设置客户列表的列名、对齐方式和宽度，然后点击下一步，显示界面：

创建数据查询TFP

第四步：设置动态查询参数

查询字段：

查询条件：

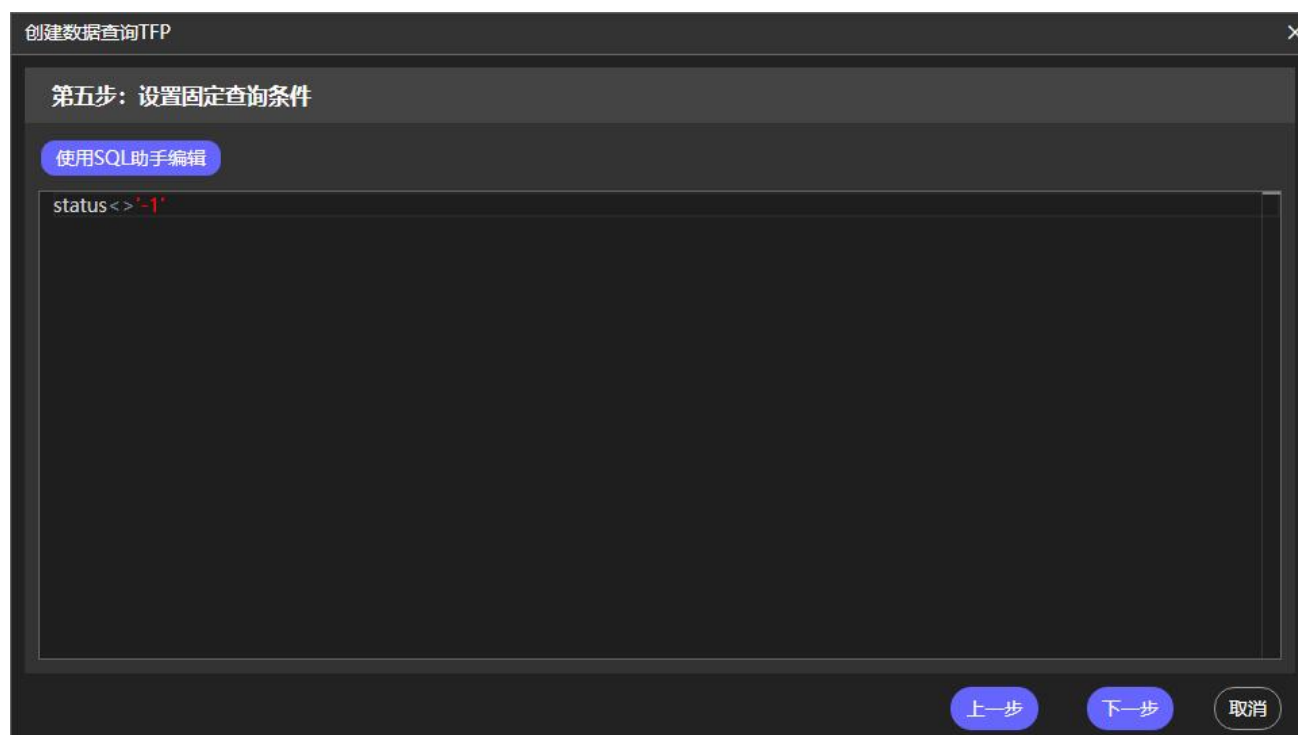
| 参数名称     | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)                    | 默认值 | 数据字典(双击选择) | 可选值 |
|----------|------|-------|----|--------------------------------|-----|------------|-----|
| code     | 编码   | 单行文本  |    | code like {%req.code%}         |     |            |     |
| name     | 名称   | 单行文本  |    | name like {%req.name%}         |     |            |     |
| contacts | 联系人  | 单行文本  |    | contacts like {%req.contacts%} |     |            |     |

上一步

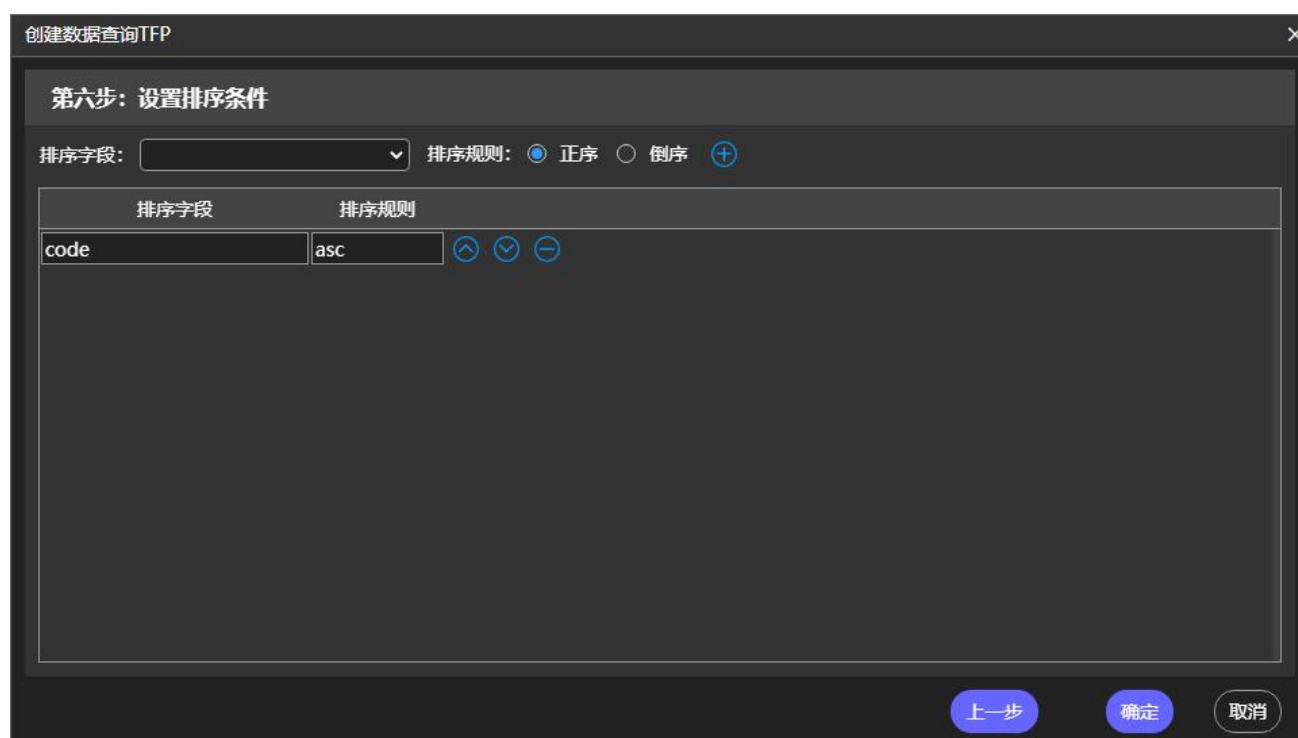
下一步

取消

在此，分别将客户编码、名称和联系人这几个字段添加为动态查询参数，查询条件都选择模糊查询，然后点击下一步，显示如下界面：



在此设置固定查询条件：status<>'1'，表示不查询已删除的客户，然后点下一步，显示如下界面：

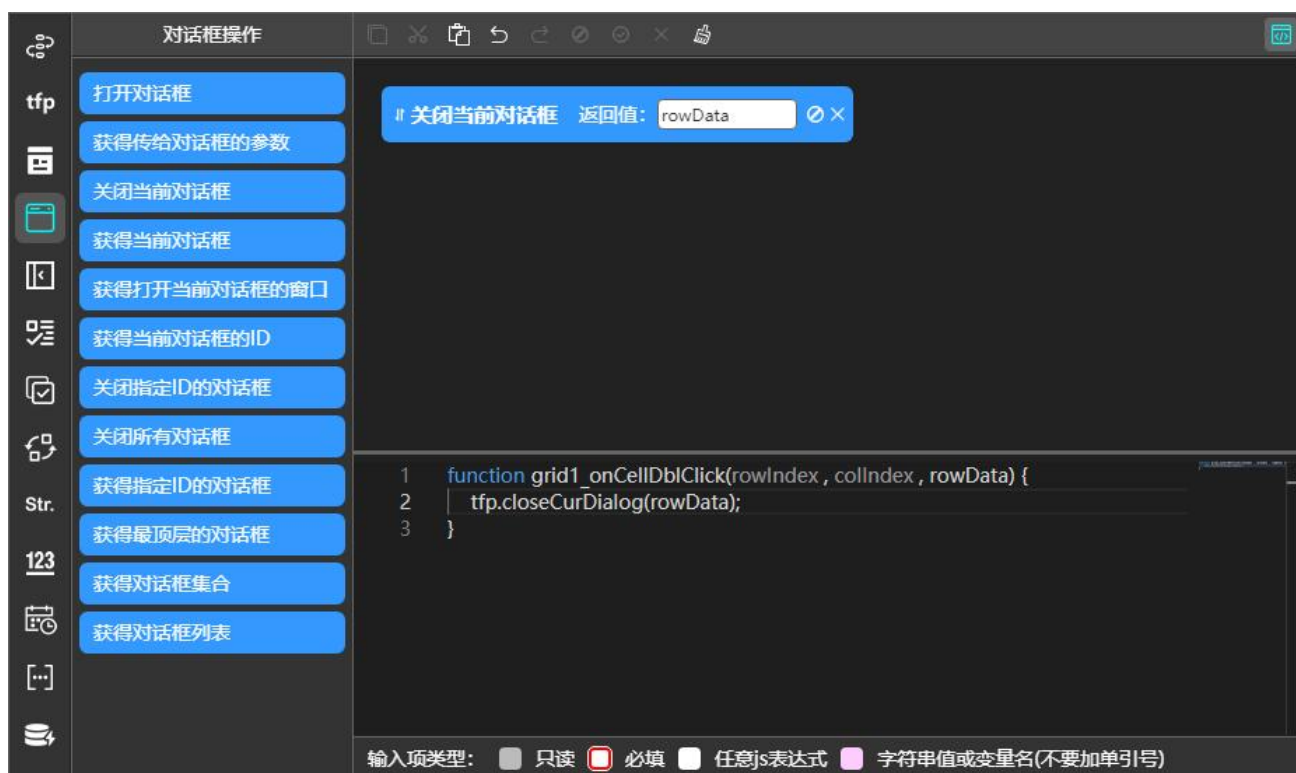


在此，将客户编码字段设置为查询条件，排序规则选择正序，然后点击确定按钮，即可完成客户列表页面的创建，如下图所示：





然后选中数据列表组件 grid1，在右侧的事件设置面板里，双击“双击单元格时”事件输入框，打开事件脚本编辑器，从对话框操作里，拖拽一条“关闭当前对话框”语句，返回值设置为：rowData，表示双击客户列表时，将双击的行对应的客户信息返回给打开该对话框的页面，并关闭该对话框。



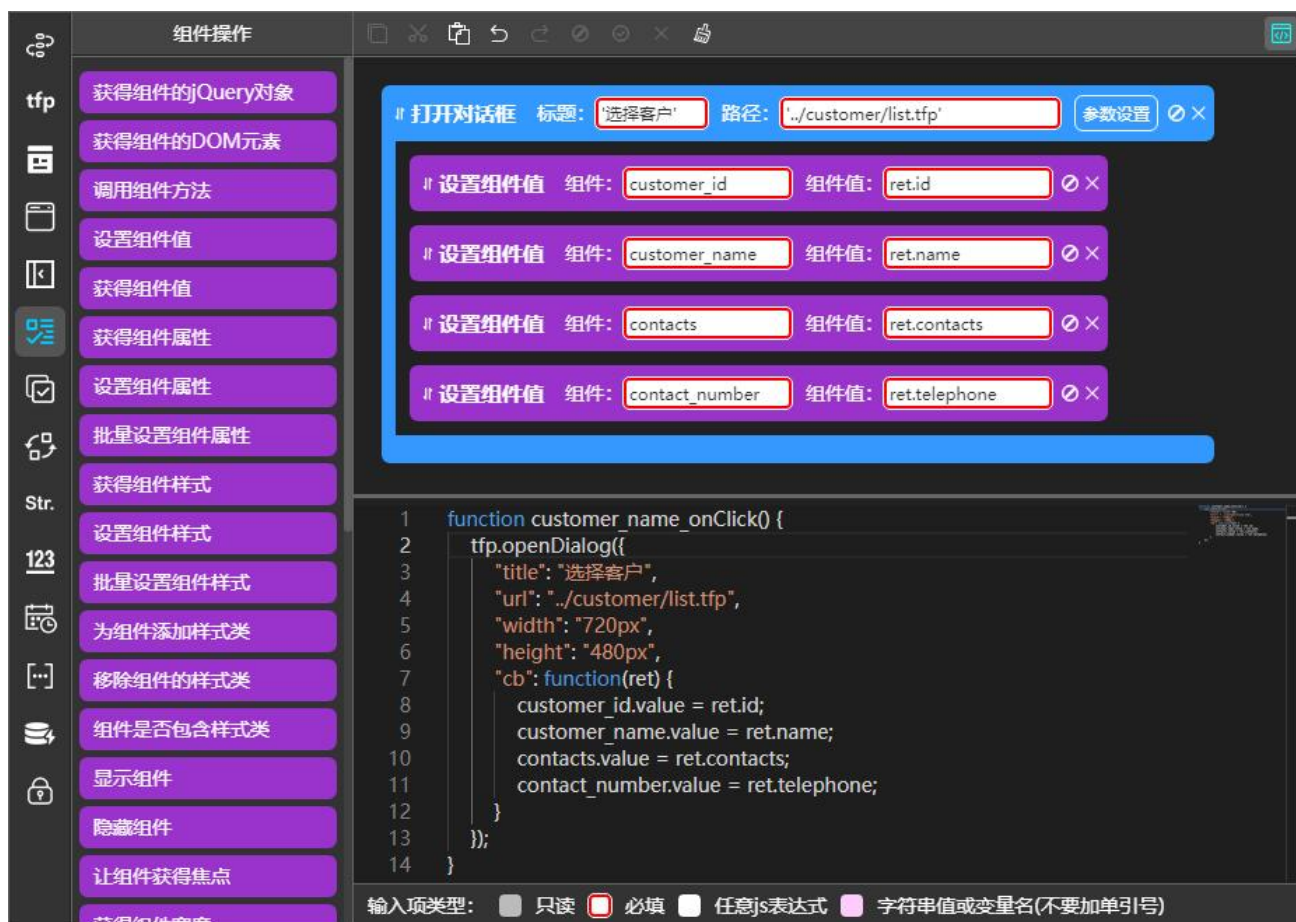
### 5.7.1.2 修改添加销售订单页面

操作步骤如下：

- (1) 选择所属客户输入项 `customer_id`，在属性设置面板中，将其组件编号改成：`customer_name`，因为这个输入项是用来显示客户名称的，`customer_id` 要分配给下面要添加的用来保存客户编号的隐藏域，这样组件的编号和里面保存的内容就能更加匹配；
- (2) 从组件库的基础输入项里，拖拽一个隐藏域到表单组件 `form1` 的底部，然后在属性设置面板中，将其组件编号改成：`customer_id`；
- (3) 选中所属客户输入项 `customer_name`，在右侧的事件设置面板里，双击“点击时”事件输入框，打开事件脚本编辑器，从对话框操作中，拖拽一个“打开对话框”语句，然后点击该语句的“参数设置”按钮，打开如下对话框，并按下图所示设置相关参数，然后点确定。



- (4) 从组件操作里，拖拽四个“设置组件值”语句，并按照下图所示设置四个组件的值：



上图的语句表示，当选择完客户后，将页面里与客户信息有关的几个输入项的值设置为弹窗选择的客户对应的信息。

这样设置好后，就可以实现弹窗选择客户，并自动输入客户有关信息，不需要手动输入。

在 TaskBuilder 里打开销售管理示例项目前端文件里 order 目录里的 index.tfp 文件，然后点预览按钮，然后在列表页面里点添加按钮打开添加销售订单的抽屉页，然后点击所示客户输入框，即可看到上面设置的客户列表对话框：

The screenshot shows a web application interface for adding a sales order. On the left is a sidebar with a '刷新' (Refresh) button and a list of orders. The main area is titled '添加' (Add) and contains a form with fields for '订单编码' (Order Code), '经办人' (Handler), '下单日期' (Order Date), '所属客户' (Customer), '联系人' (Contact), '联系电话' (Contact Phone), '订单金额' (Order Amount), '优惠金额' (Discount Amount), and '实收金额' (Actual Amount). A '备注' (Remarks) field is at the bottom. A modal window titled '选择客户' (Select Customer) is open, displaying a table with customer information.

| 客户编码  | 客户名称             | 联系人 | 联系电话         |
|-------|------------------|-----|--------------|
| 10011 | 深圳某电气设备生产制造有限公司1 | 李总1 | 136789054321 |

The modal also includes search fields for '编码' (Code), '名称' (Name), and '联系人' (Contact), a '查询' (Search) button, and pagination controls showing '共 1 页, 1 条记录' (Total 1 page, 1 record).

在列表里双击某个客户，即可将该客户的信息自动录入到添加销售订单页面里：

This screenshot shows the same '添加' (Add) form, but now the '所属客户' (Customer), '联系人' (Contact), and '联系电话' (Contact Phone) fields are pre-filled with data from the modal in the previous screenshot. These three fields are highlighted with a red rectangular box. The '订单编码' (Order Code) field contains 'XS2024080004'. The '经办人' (Handler) field is empty. The '下单日期' (Order Date) field is empty. The '订单金额' (Order Amount), '优惠金额' (Discount Amount), and '实收金额' (Actual Amount) fields are empty. The '备注' (Remarks) field is empty.

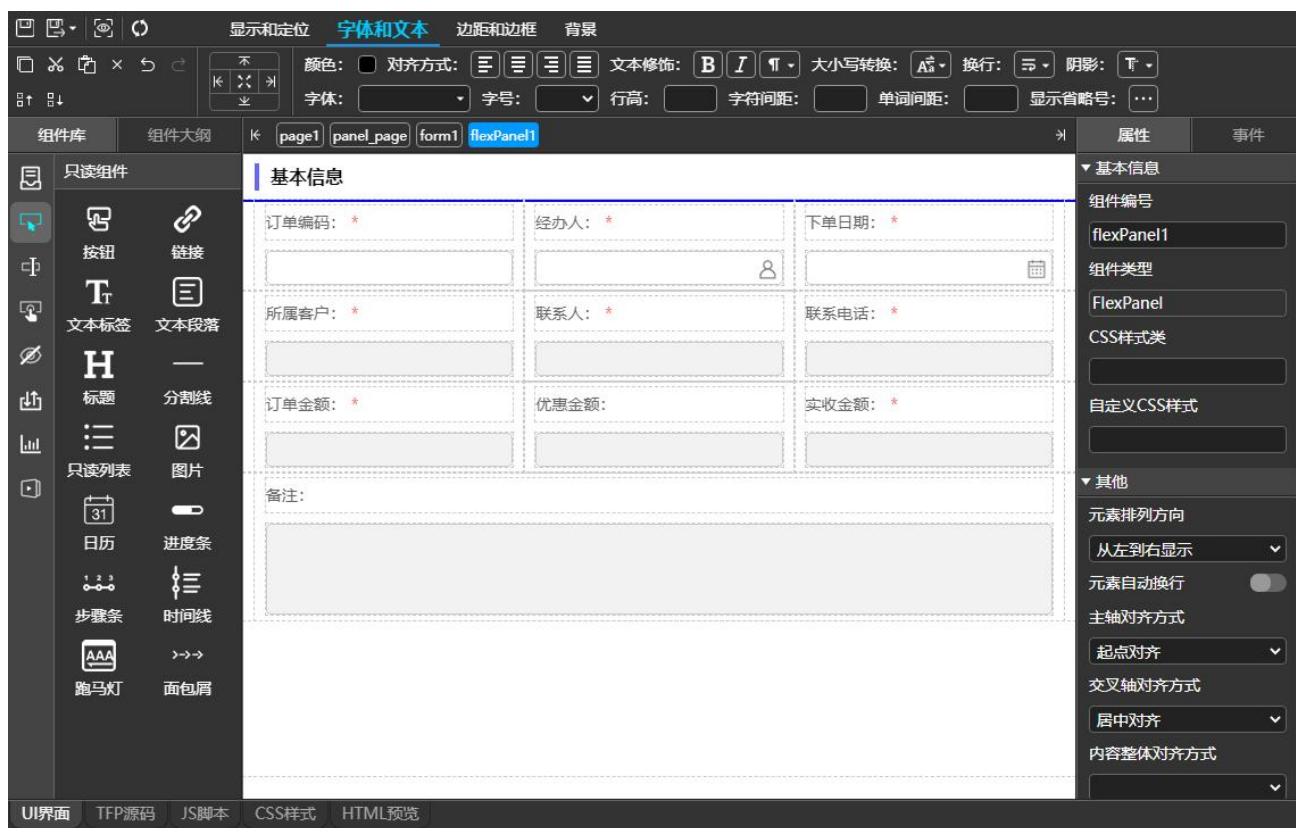
## 5.7.2 实现内容分栏显示

操作步骤如下：

- (1) 在 UI 设计器中，选中添加销售订单页面里的表单组件 form1；
- (2) 从组件库的容器组件中，拖拽一个弹性面板组件到表单组件的底部；然后在上部的样式设置栏中，将这个弹性面板的高度设置为 40px；
- (3) 从组件库的只读组件中，拖拽一个文本标签组件到上面的弹性面板组件里，在右侧的属性设置面板里，将其组件文本设置为：基本信息，附加内容选择：竖条前缀；然后在上部的样式设置栏的字体和文本中，将字号设置为：16px，并且加粗；
- (4) 选中上面这个文本标签所在的弹性面板组件，点击工具栏上的“移到前面”按钮：

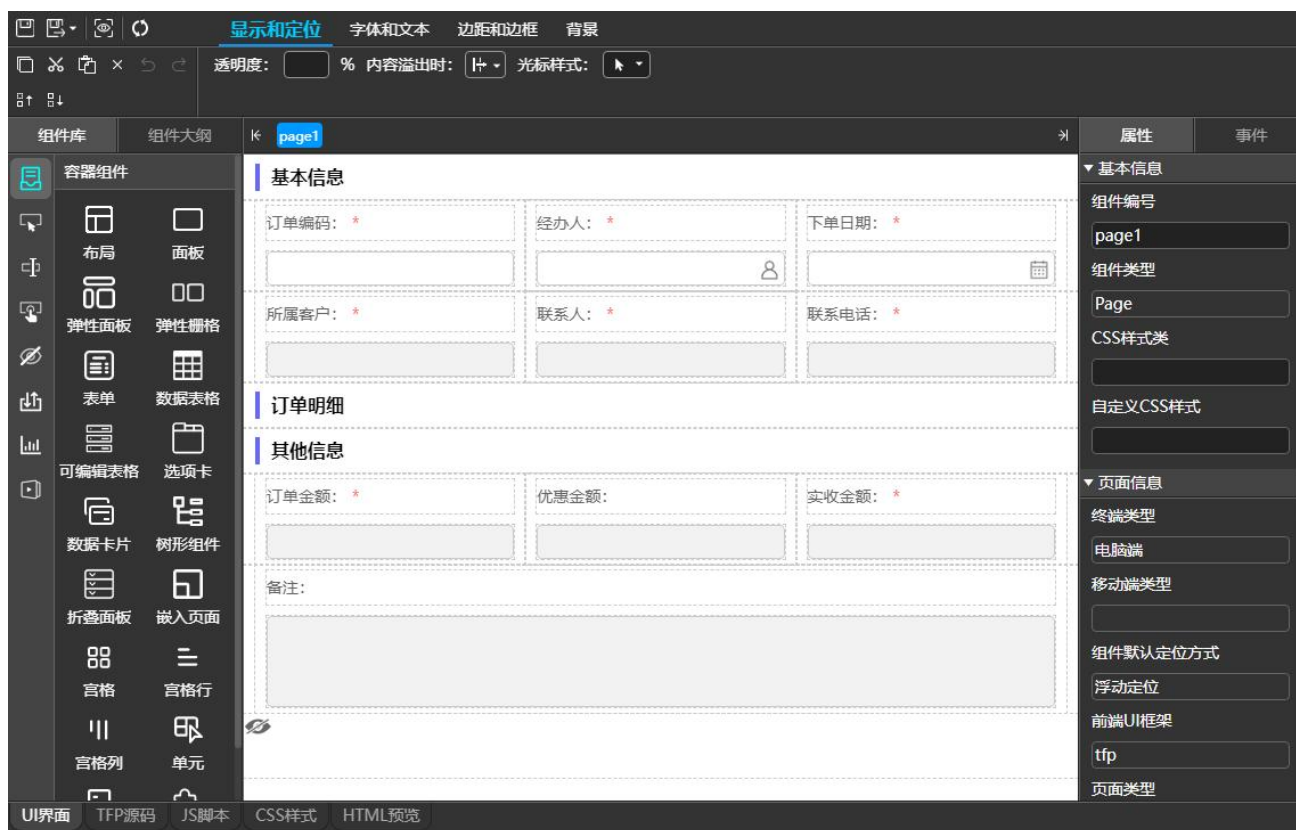


将该面板组件往上移动，一直移动到表单组件的最上面一行，移动完的界面如下所示：



参考上面的操作，再添加订单明细和其他信息这两个标题栏，添加完的界面如下：





这样设置后，界面里的内容分成几个区域显示，看起来会更加美观和清晰。

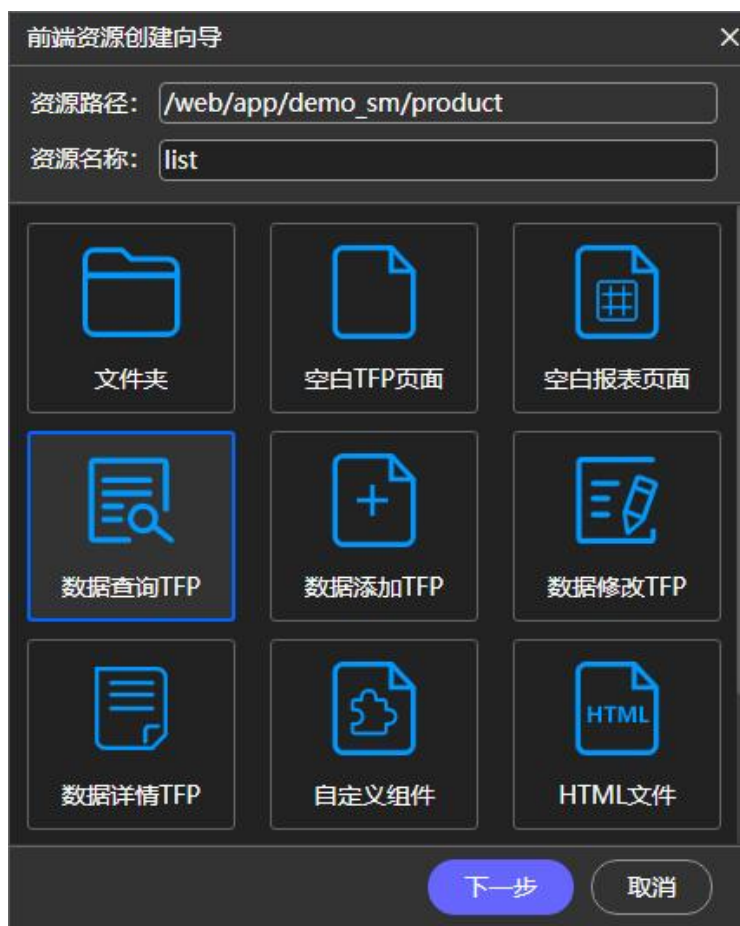
### 5.7.3 实现订单明细数据录入

该功能需要分以下两步实现：

#### 5.7.3.1 创建选择产品信息对话框

在销售管理示例项目的前端文件中，点击 product 文件夹右侧的加号按钮，打开“前端资源创建向导”对话框，在该对话框中，资源类型选择“数据查询 TFP”，资源名称设置为 list，界面如下：





设置好后，点击下一步按钮，显示如下所示的“创建数据查询 TFP”对话框：



在此对话框中，数据模型选择：demo\_sm\_product.tdm，页面标题设置为：选择产品，页

面类型设置为对话框，并取消选中显示添加按钮、显示修改按钮、显示删除按钮这三个选项，然后点下一步，显示如下界面：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段 添加关联字段

| 字段名称                                      | 字段备注   | 字段格式(双击编辑) | 关联数据字典(双击选择) | 字段别名       |
|-------------------------------------------|--------|------------|--------------|------------|
| <input checked="" type="checkbox"/> id    | 产品编号   | id         |              | product_id |
| <input checked="" type="checkbox"/> code  | 产品编码   | code       |              |            |
| <input checked="" type="checkbox"/> name  | 产品名称   | name       |              |            |
| <input type="checkbox"/> type_id          | 产品分类编号 | type_id    |              |            |
| <input type="checkbox"/> sort_num         | 产品序号   | sort_num   |              |            |
| <input checked="" type="checkbox"/> model | 规格型号   | model      |              |            |
| <input checked="" type="checkbox"/> unit  | 计量单位   | unit       | unit         |            |
| <input checked="" type="checkbox"/> price | 单价     | price      |              |            |
| <input type="checkbox"/> thumbnail        | 缩略图代码  | thumbnail  |              |            |
| <input type="checkbox"/> comment          | 备注     | comment    |              |            |
| <input type="checkbox"/> status           | 状态     | status     |              |            |
| <input type="checkbox"/> deleter_id       | 删除人编号  | deleter_id |              |            |

上一步 下一步 取消

在此选中 id、code、name、model、unit、price 这几个字段，并且将 id 字段的别名设置为 product\_id，unit 字段设置关联数据字典为 unit。然后再添加一个关联字段，设置界面如下：

添加关联字段

关联对象类型：  
☒ 表 ☐ 视图

关联数据对象：  
demo\_sm\_product\_type

关联到主表字段：  
name

关联字段别名：  
type\_name

关联字段备注：  
分类名称

关联表字段：  
id  
code  
name  
parent\_id  
sort\_num  
status

主表字段：  
id  
code  
name  
type\_id  
sort\_num  
model  
unit  
price  
thumbnail  
comment

关联条件：  
and or = <> > < >= <= like in 为空 不为空

id = demo\_sm\_product.type\_id

确定 取消

然后再点下一步，显示如下界面：

创建数据查询TFP

第三步：设置显示格式

| 列名   | 列类型 | 数据绑定格式(js表达式) | 文本显示格式 | 对齐方式 | 宽度  | 文字颜色 | 加粗 | 求和 |
|------|-----|---------------|--------|------|-----|------|----|----|
| 分类名称 | 文本  | {type_name}   |        | 居中对齐 | 80  |      |    |    |
| 产品编码 | 文本  | {code}        |        | 居中对齐 | 80  |      |    |    |
| 产品名称 | 文本  | {name}        |        | 居中对齐 | 150 |      |    |    |
| 规格型号 | 文本  | {model}       |        | 居中对齐 | 230 |      |    |    |
| 单位   | 文本  | {unit}        |        | 居中对齐 | 60  |      |    |    |
| 单价   | 文本  | {price}       |        | 居中对齐 | 80  |      |    |    |

上一步 下一步 取消

在此，先删除产品编码，不显示这个字段，然后按上图所示设置各个字段的显示格式，然后再点下一步，显示如下界面：

创建数据查询TFP

第四步：设置动态查询参数

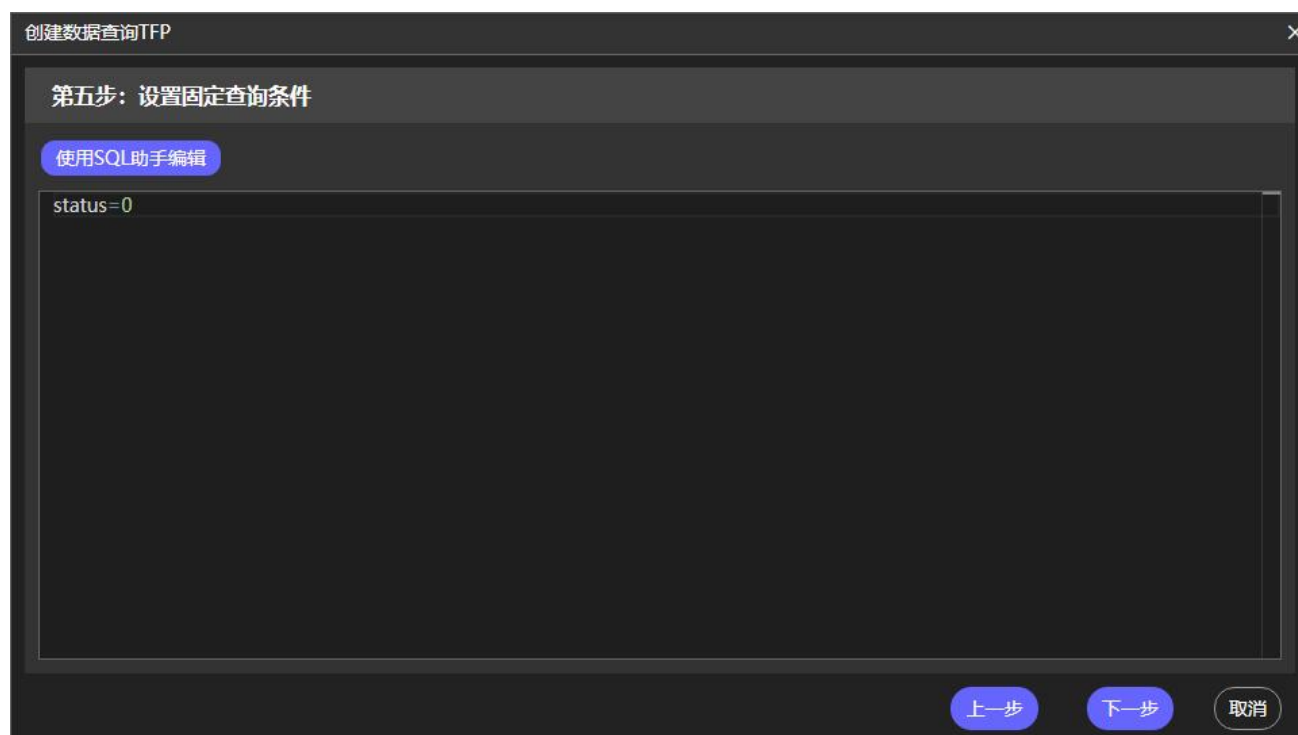
查询字段：

查询条件：

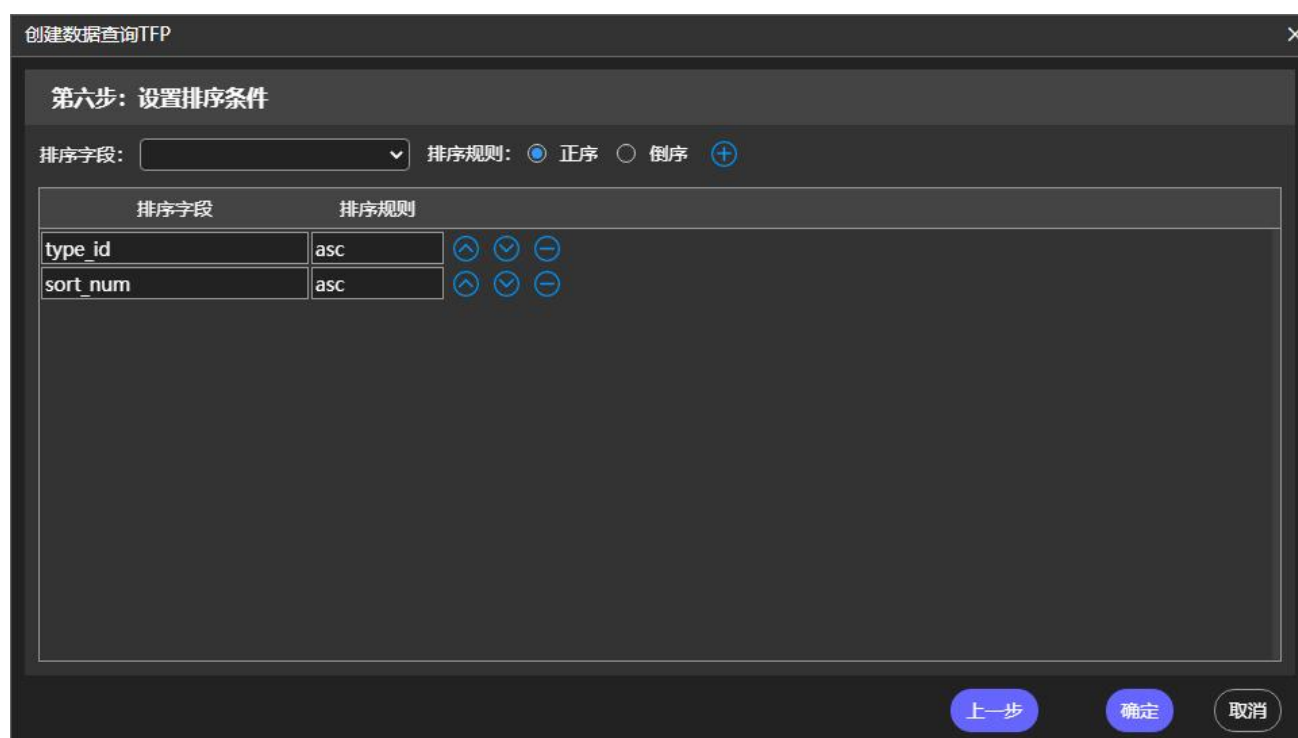
| 参数名称    | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)            | 默认值 | 数据字典(双击选择) | 可选值 |
|---------|------|-------|----|------------------------|-----|------------|-----|
| code    | 产品编码 | 单行文本  |    | code like {%req.code%} |     |            |     |
| name    | 产品名称 | 单行文本  |    | name like {%req.name%} |     |            |     |
| type_id | 产品分类 | 下拉列表  |    | type_id={req.type_id}  |     |            |     |

上一步 下一步 取消

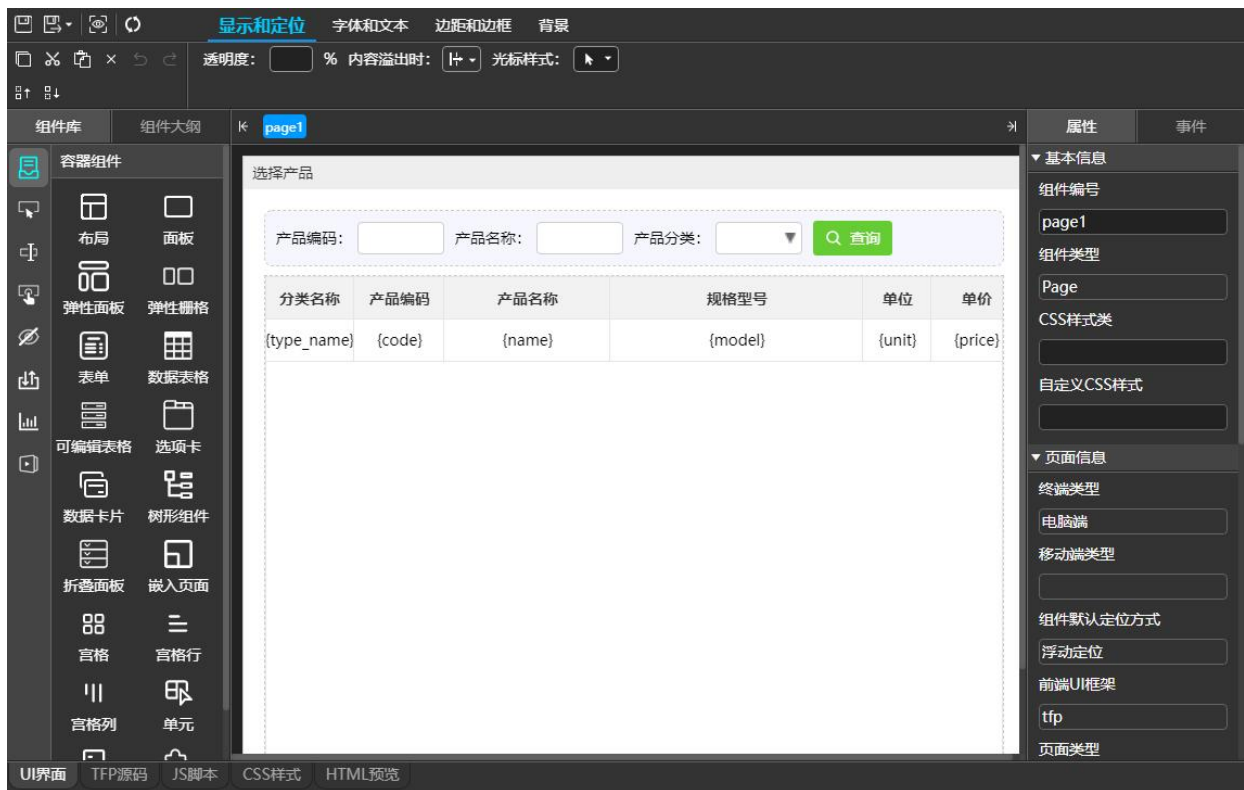
在此按照上图所示添加三个动态查询参数，然后继续点击下一步，显示下图：



在此将固定查询条件设置为：status=0，表示仅查询状态为正常的产品，然后再点下一步：



在此，参考上图添加两个排序条件，然后点确定，即可创建该页面，如下图所示：



### 5.7.3.2 设置产品分类下拉列表

在该页面里，选中查询条件面板里的产品分类下拉列表组件，先将其宽度设置为 120px，然后点击属性里的数据查询设置，打开如下对话框：



在此对话框中，先选择数据模型 demo\_sm\_product\_type.tdm，然后选择字段 id、name 和 parent\_id，然后点击查询条件选项卡，将固定查询条件设置为：status=0，如下图所示：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 查询条件 分组和排序

固定查询条件 查询字段:  查询条件:  (+)

status=0

动态查询条件 查询字段:  查询条件:  (+)

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|

清空 确定 取消

然后点分组和排序选项卡，按下图所示设置排序条件：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型 请求参数 查询条件 分组和排序

分组条件 分组字段:  (+)

排序条件 字段:  正序 (+)

| 排序字段      | 排序规则 |       |
|-----------|------|-------|
| parent_id | asc  | ^ v - |
| sort_num  | asc  | ^ v - |

分组过滤条件 过滤字段:  过滤条件:  (+)

清空 确定 取消



最后点确定，完成设置。然后在产品分类下拉列表组件的属性设置里，设置以下属性：

- **选项值字段**：设置为 id；
- **选项文本字段**：设置为 name；
- **显示树形结构**：启用；
- **父节点主键字段**：设置为 parent\_id；
- **根节点主键值**：设置为 0；

都设置完后，打开该页面时，产品分类下拉列表就会按树形结构显示所有产品分类。

### 5.7.3.3 设置产品列表数据表格

在选择产品页面中，选中产品列表数据表格组件 grid1，在属性设置的表体设置里，将“显示选择框”选项选中，这样就可以在产品列表前面显示选择框，以便选择一个或多个产品。



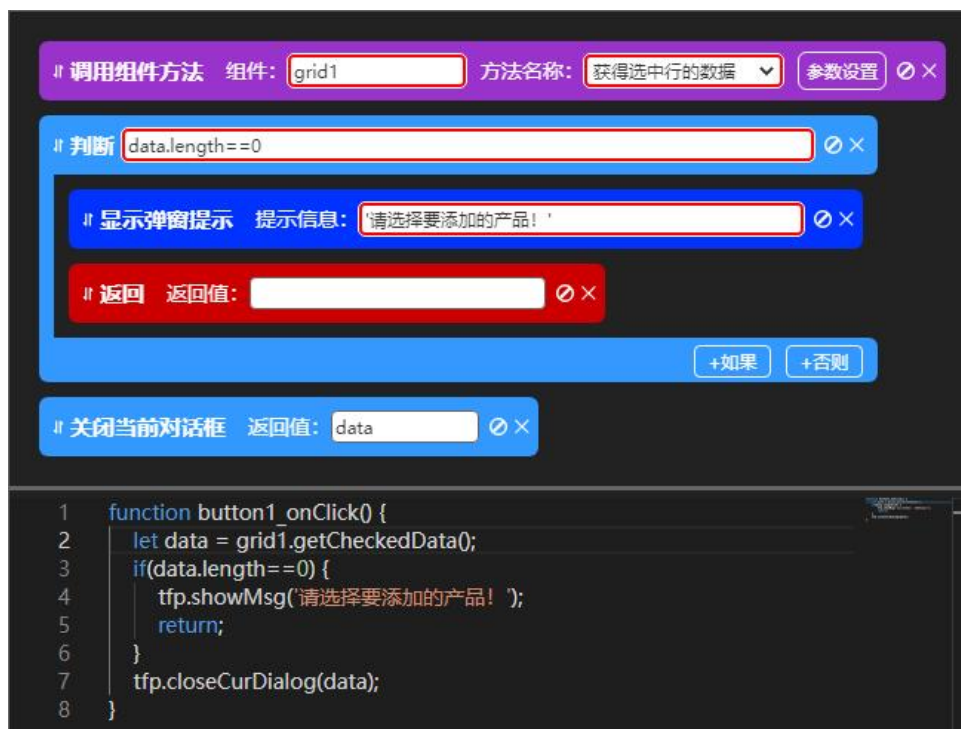
### 5.7.3.4 添加和设置确定按钮

从组件库的只读组件中，拖拽一个按钮到查询按钮后面，并在上面的样式设置栏的边距和边框设置里，将外边距的左边距设置为 15px，然后在属性设置里设置以下属性：

- **文本内容**：设置为确定；
- **按钮类型**：设置为主要按钮；
- **显示图标**：启用；
- **图标**：设置为确定；

然后在该按钮的事件设置里，双击“点击时”事件输入框，打开事件脚本设计器，按下图

所示设置相关语句：



以上语句说明如下：

- (1) 先从语句库的组件操作中，拖拽一个调用组件方法语句，组件设置为 grid1，方法名称选择获得选中行的数据，该语句默认会将返回数据存储在名为 data 的变量中；
- (2) 从基本语句里拖拽一个判断语句，判断条件设置为 data.length==0，表示如果没有选中任何行，就执行判断语句里的子语句；
- (3) 从 tfp 框架基本功能里拖拽一个显示弹窗提示语句到判断语句里，提示信息设置为：'请选择要添加的产品！'，注意：提示内容两边要加单引号，表示是字符串，不是变量；
- (4) 再从基本语句里拖拽一个返回语句到判断语句里，表示提示完就退出事件函数，不再执行后续的语句；
- (5) 从对话框操作中拖拽一个关闭当前对话框语句，将返回值设置为 data，表示关闭对话框的同时，将选中的产品信息传给打开该对话框的页面；

以上内容都设置好后，该页面就可以给添加销售订单的页面使用了。

### 5.7.3.5 添加可编辑表格

在 TaskBuilder 内，打开前面创建的添加销售订单页面，从组件库的容器组件里，拖拽一个可编辑表格组件到表单组件底部，然后点击工具栏上的“移到前面”按钮 ，将该组件移动到订单明细标题栏下面，如下图所示：



然后在上面样式设置栏的显示和定位里，将定位方式改为流式定位，如下图所示：



然后在边距和边框中，将外边距的左边距和右边距都设置为 20，如下图所示：



这样设置后，可编辑表格左右两侧会保留 20 像素的边距，跟其他行的输入项边距保持一致，界面会更加美观。

### 5.7.3.6 设置可编辑表格的数据列

在上面添加的可编辑表格的属性设置中，点击数据列设置，打开可编辑表格的列设置对话框，把原来默认创建的三个列删除，然后按照下表所示添加和设置新的列：

| 列 ID       | 列名   | 数据类型 | 宽度  | 默认值  | 其他                            |
|------------|------|------|-----|------|-------------------------------|
| product_id | 产品编号 | 文本   | 60  |      | 隐藏，不显示该列，但提交时需要               |
| name       | 产品名称 | 文本   | 120 |      | 只读                            |
| model      | 规格型号 | 文本   | 240 |      | 只读                            |
| price      | 单价   | 金额   | 80  | 0.00 | 必填                            |
| quantity   | 数量   | 整数   | 60  | 0    | 必填                            |
| unit       | 单位   | 文本   | 50  |      | 只读                            |
| amount     | 金额   | 金额   | 80  | 0.00 | 只读，自动计算表达式：{price}*{quantity} |
| comment    | 备注   | 文本   | 130 |      |                               |

数据列设置界面如下所示：

设置数据集

| 列 ID       | 列名   | 数据类型 | 宽度  | 默认值  | 其他                            |
|------------|------|------|-----|------|-------------------------------|
| product_id | 产品编号 | 文本   | 60  |      | 隐藏，不显示该列，但提交时需要               |
| name       | 产品名称 | 文本   | 120 |      | 只读                            |
| model      | 规格型号 | 文本   | 240 |      | 只读                            |
| price      | 单价   | 金额   | 80  | 0.00 | 必填                            |
| quantity   | 数量   | 整数   | 60  | 0    | 必填                            |
| unit       | 单位   | 文本   | 50  |      | 只读                            |
| amount     | 金额   | 金额   | 80  | 0.00 | 只读，自动计算表达式：{price}*{quantity} |
| comment    | 备注   | 文本   | 130 |      |                               |

配置选项：

- 列ID: amount
- 列名: 金额
- 输入项类型: 单行输入框
- 数据类型: 金额
- 组织结构数据类型: 编号
- 宽度: 80
- 默认值: 0.00
- 只读: ☒
- 必填: ☐
- 文本验证: (表达式)
- 隐藏: ☐
- CSS样式:
- 自动计算表达式: {price}\*{quantity}
- CSS样式类:

确定 取消

### 5.7.3.7 设置可编辑表格的弹窗选择参数

选中可编辑表格，在属性里设置以下参数：

- **显示添加按钮：**关闭；
- **显示从弹窗选择添加：**打开；
- **弹窗选择按钮文本：**选择产品；
- **弹窗页面路径：** /app/demo\_sm/product/list.tfp；
- **弹窗宽度：** 800；
- **弹窗高度：** 540；

设置好这些参数后，预览添加销售订单页面，就可以看到可编辑表格上方有一个名为“选择产品”的按钮，点击该按钮，即可打开选择产品对话框，选择多个产品，点确定即可添加。

### 5.7.3.8 设置可编辑表格的数据提交参数

选中可编辑表格，点击属性设置里的数据提交设置，打开如下对话框：



数据提交设置

提交数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据模型** 请求参数

提交类型：☒ 添加 ☐ 修改 ☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

| ■                                   | 字段名称       | 字段备注 | 字段值           | 主键                         |
|-------------------------------------|------------|------|---------------|----------------------------|
| <input checked="" type="checkbox"/> | order_id   | 订单编号 | ret.insertId  | <input type="checkbox"/> - |
| <input checked="" type="checkbox"/> | product_id | 产品编号 | #{product_id} | <input type="checkbox"/> - |
| <input checked="" type="checkbox"/> | price      | 单价   | #{price}      | <input type="checkbox"/> - |
| <input checked="" type="checkbox"/> | quantity   | 数量   | #{quantity}   | <input type="checkbox"/> - |
| <input checked="" type="checkbox"/> | unit       | 计量单位 | #{unit}       | <input type="checkbox"/> - |
| <input checked="" type="checkbox"/> | amount     | 金额   | #{amount}     | <input type="checkbox"/> - |
| <input checked="" type="checkbox"/> | comment    | 备注   | #{comment}    | <input type="checkbox"/> - |

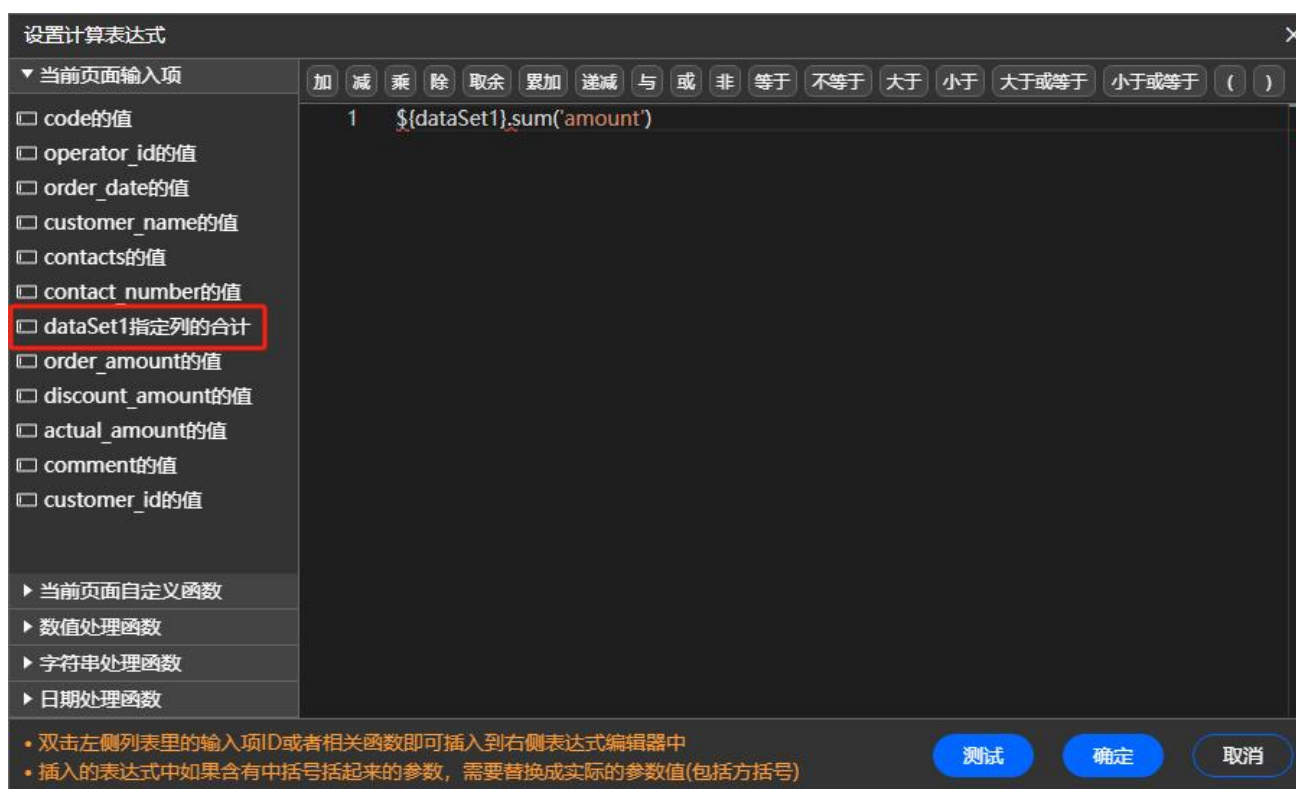
在该对话框中，按照上图所示选中所有字段，并设置各个字段的值，然后点确定，即可完成可编辑表格的数据提交设置，然后再保存页面，既可以在该页面对应的数据提交的后台服务中自动生成保存订单明细数据的语句，不需要再手动设置。

## 5.7.4 实现金额自动计算

接下来，我们实现页面内几个金额输入项的自动计算功能，操作步骤如下：

### 5.7.4.1 设置订单金额输入项的自动计算表达式

在添加销售订单页面里选中“订单金额”输入项（order\_amount），然后点击属性设置里的“自动计算表达式”，打开如下图所示对话框：



在该对话框中，双击左侧列表里的“dataSet1 指定列的合计”，将该函数插入到右边的计算表达式设计器里，然后将里面的[列 ID]改成 amount，表示该页面里的“订单金额”输入项的值是可编辑表格里的金额（列 ID：amount）这一列所有单元格值的合计，当可编辑表格里任意一行的金额发生变化时，就会自动计算这一列的合计值，并更新“订单金额”输入项的值，

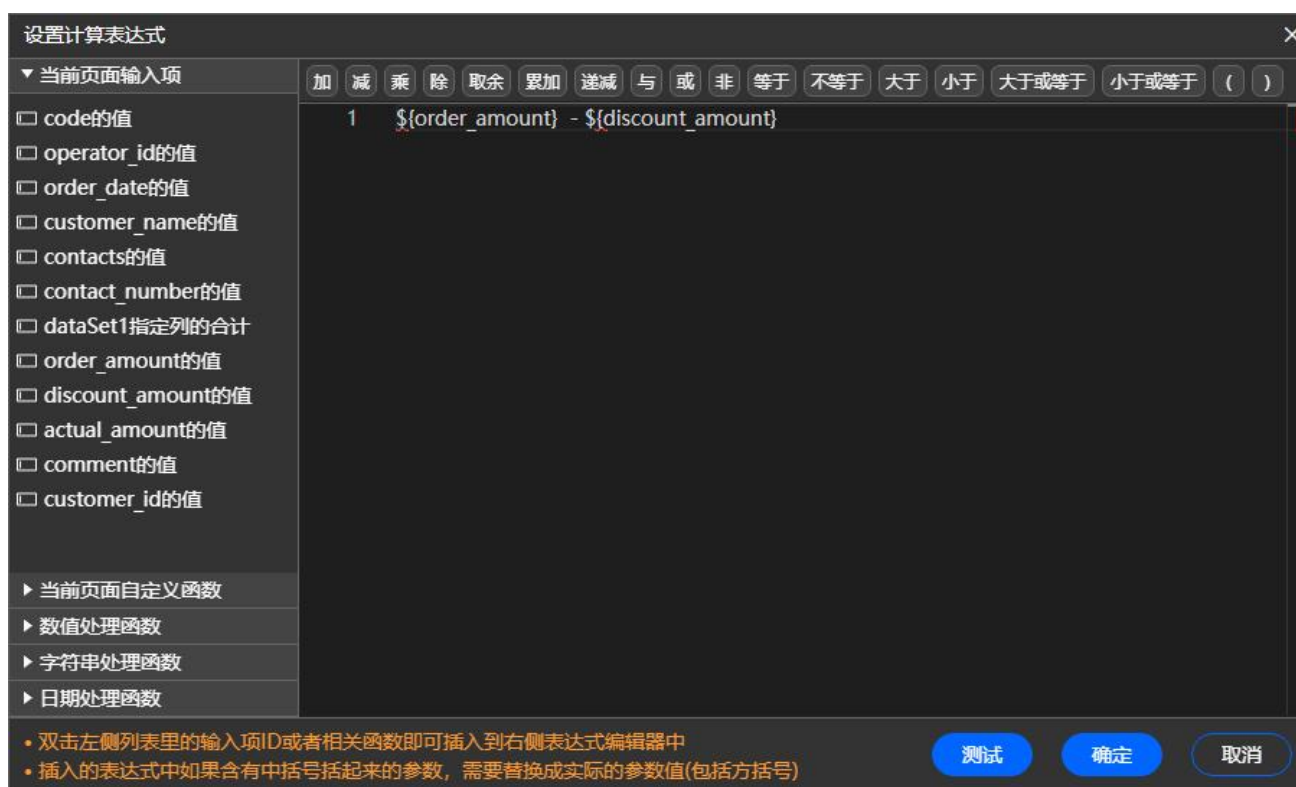


设置好后，点击确定，完成设置。

另外，还需要在属性设置里将“订单金额”输入项设置为只读，因为这个输入项的值是自动计算出来的，不需要用户输入。

#### 5.7.4.2 设置实收金额输入项的自动计算表达式

在添加销售订单页面里选中“实收金额”输入项（actual\_amount），然后点击属性设置里的“自动计算表达式”，打开如下图所示对话框：



在此，依次双击左侧列表里的“order\_amount 的值”和“discount\_amount 的值”，然后在两个值之间添加减号，这样就表示“实收金额”输入项的值 = 订单金额 - 优惠金额，设置好后，点击确定，完成设置。

另外，还需要在属性设置里将“实收金额”输入项设置为只读，因为这个输入项的值是自动计算出来的，不需要用户输入。

还有，选中“优惠金额”输入项（discount\_amount），在属性里，将数据类型设置为金额，这样设置后，当用户输入完优惠金额后，会自动格式化为金额格式。

## 5.7.5 设置输入项的默认值

为了便于用户操作，可以为页面里的一些输入项设置一些默认值：

### 5.7.5.1 设置经办人默认值

在添加销售订单页面里选中“经办人”输入项（operator\_id），在属性里将“默认为当前用户”开关打开。

### 5.7.5.2 设置下单日期默认值

在添加销售订单页面里选中“下单日期”输入项（order\_date），在属性里将“默认为当前时间”开关打开。

## 5.8 预览测试添加销售订单页面

至此，添加销售订单页面就全部设置完毕，就可以预览列表页面，然后点击添加按钮进行销售订单的添加操作测试。

刷新

+

添加

订单编码:

经办人:

| 订单编码         | 经办人 | 下单日期       | 客      |
|--------------|-----|------------|--------|
| XS2024080023 | 马路  | 2024-08-17 | 深圳某电气设 |
| XS2024080019 | 刘鑫  | 2024-08-10 | 深圳某电气设 |

添加

确定取消

基本信息

订单编码: \*

经办人: \*

下单日期: \*

所属客户: \*

联系人: \*

联系电话: \*

订单明细

选择产品

| 产品名称 | 规格型号        | 单价     | 数量 | 单位 | 金额      | 备注 |
|------|-------------|--------|----|----|---------|----|
| 电热水壶 | 澳柯玛1L不锈钢电水壶 | 120.00 | 10 | 台  | 1200.00 |    |
| 拖把   | 大卫免手洗平板拖把   | 71.90  | 10 | 吨  | 719.00  |    |

其他信息

订单金额: \*

优惠金额:

实收金额: \*

备注:

## 5.9 创建修改销售订单页面

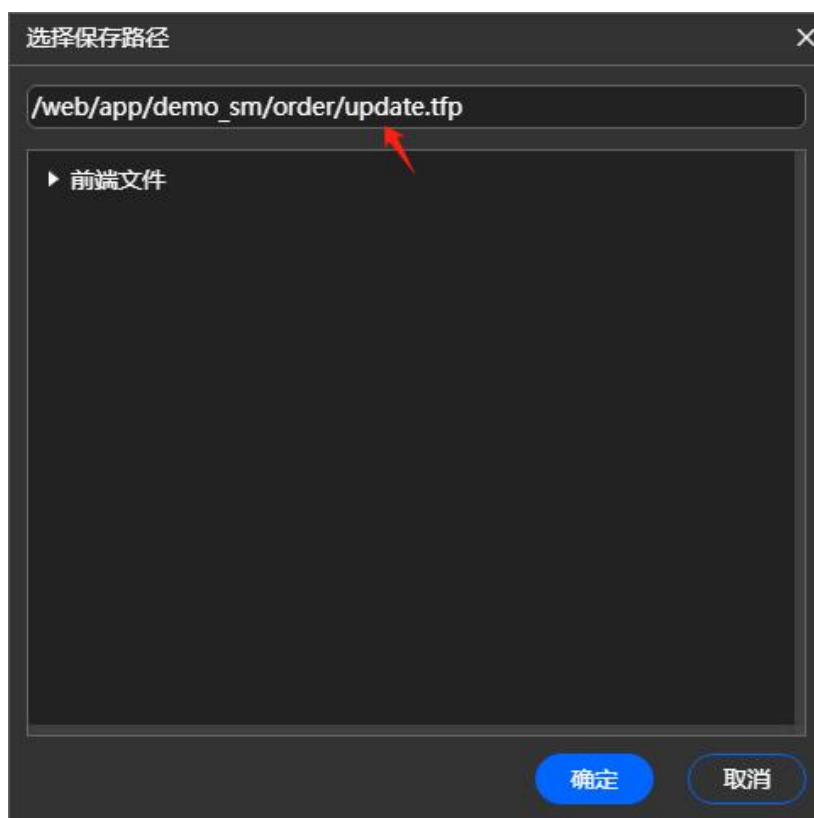
修改销售订单页面和添加页面的内容基本一样，所以我们可以走一下捷径，直接把添加订单页面另存为修改销售订单页面，然后再修改完善一下即可，具体操作如下：

### 5.9.1 另存添加销售订单页面

打开添加销售订单页面，然后点击工具栏里的另存为按钮，在弹出的菜单了选择“另存”：



在弹出的对话框中，将文件名改为 update.tfp，如下图所示：



然后点确定，即可将添加销售订单页面另存为查看订单详情页面，这样就不用从头构建这个页面，只需要做一些完善即可。

## 5.9.2 修改页面标题

在另存的这个页面里，选中页面组件（page1），在属性设置里，将标题修改为：修改销售单，然后保存页面。

## 5.9.3 更换订单编码输入项

在修改订单信息页面里，订单编码不需要再重新生成，只需要显示在添加订单时已经生成的订单编码即可，所以不能再使用编码规则组件，需要更换为单行输入框，操作方法如下：

- （1）选择订单编码对应的编码规则组件 code，然后点击工具栏上的删除按钮 ，即可删除；
- （2）从组件库的基础输入项里，拖拽一个单行输入框到上面这个删除的编码规则组件所在的弹性栅格组件里，并在属性设置里，将组件编号设置为 code，与原来的保持一致；并设置为只读，不允许修改；另外，在上面的样式设置栏里，将宽度设置为 100%。

## 5.9.4 设置表单数据查询参数

在修改页面里，需要先查询需要修改的数据，并显示在各个输入项中，设置方法如下：

选中表单组件（form1），点击属性里的数据查询设置，会打开下图所示的对话框：

在此，先选择销售订单数据模型：demo\_sm\_sale\_order.tdm，然后全选所有字段，然后添加以下关联字段，关联客户信息表，根据客户编号查询客户的名称：

| 关联表              | 关联字段 | 字段别名          | 字段备注 | 关联条件                                |
|------------------|------|---------------|------|-------------------------------------|
| demo_sm_customer | name | customer_name | 客户名称 | id = demo_sm_sale_order.customer_id |

数据查询设置

查询数据方式: ☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

**数据模型** 请求参数 查询条件 分组和排序

☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径:

| 字段名称                                                | 字段备注  | 字段格式(SQL, 双击可编辑) | 字段别名 | 数据字典(双击选择) |
|-----------------------------------------------------|-------|------------------|------|------------|
| <input checked="" type="checkbox"/> id              | 订单编号  |                  |      |            |
| <input checked="" type="checkbox"/> code            | 订单编码  |                  |      |            |
| <input checked="" type="checkbox"/> operator_id     | 经办人编号 |                  |      |            |
| <input checked="" type="checkbox"/> order_date      | 下单日期  |                  |      |            |
| <input checked="" type="checkbox"/> customer_id     | 客户编号  |                  |      |            |
| <input checked="" type="checkbox"/> contacts        | 联系人   |                  |      |            |
| <input checked="" type="checkbox"/> contact_number  | 联系电话  |                  |      |            |
| <input checked="" type="checkbox"/> order_amount    | 订单金额  |                  |      |            |
| <input checked="" type="checkbox"/> discount_amount | 优惠金额  |                  |      |            |
| <input checked="" type="checkbox"/> actual_amount   | 实收金额  |                  |      |            |
| <input checked="" type="checkbox"/> creator_id      | 创建人编号 |                  |      |            |

关联查询客户名称的设置界面如下:

添加关联字段

关联对象类型: ☒ 表 ☐ 视图

关联数据对象:

关联到主表字段:

关联字段别名:

关联字段备注:

关联表字段:

|             |
|-------------|
| 双击插入到关联条件中  |
| id          |
| code        |
| name        |
| creator_id  |
| create_time |
| status      |
| salesman_id |
| contacts    |
| telephone   |
| scale       |

主表字段:

|                 |
|-----------------|
| 双击插入到关联条件中      |
| id              |
| code            |
| operator_id     |
| order_date      |
| customer_id     |
| contacts        |
| contact_number  |
| order_amount    |
| discount_amount |
| actual_amount   |

关联条件:

查询字段都设置好后, 点上面数据查询设置对话框里的“请求参数”选项卡, 在此设置列

表页面传递给这个修改页面的 id 参数，界面如下：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '请求参数' (Request Parameters) tab selected. The '查询数据方式' (Query Data Method) is set to '数据模型' (Data Model). The '参数名' (Parameter Name) is 'id', the '参数类型' (Parameter Type) is 'URL 参数' (URL Parameter), and the '参数值' (Parameter Value) is empty. There are four action buttons: '+', '^', 'v', and '-'. A note at the top right says '↓ 在此设置需要传递给后台的查询参数' (↓ Set the query parameters to be passed to the backend here). At the bottom are '清空' (Clear), '确定' (Confirm), and '取消' (Cancel) buttons.

| 参数名 | 参数类型   | 参数值 |
|-----|--------|-----|
| id  | URL 参数 |     |

然后再点“查询条件”选项卡，在此设置要修改的数据的查询条件，如下图所示：

The screenshot shows the '数据查询设置' (Data Query Settings) dialog box with the '查询条件' (Query Conditions) tab selected. The '查询数据方式' (Query Data Method) is still '数据模型' (Data Model). The '固定查询条件' (Fixed Query Conditions) section has '查询字段' (Query Field) set to 'id' and '查询条件' (Query Condition) set to '{req.id}'. The '动态查询条件' (Dynamic Query Conditions) section is empty. At the bottom are '清空' (Clear), '确定' (Confirm), and '取消' (Cancel) buttons.

| 参数名称 | 参数说明 | 查询表达式(SQL, 双击可编辑) |
|------|------|-------------------|
|------|------|-------------------|



然后点确定按钮，即可完成表单数据查询的设置。

### 5.9.5 设置表单输入项数据绑定表达式

通过上面的表单数据查询设置，可以将销售订单主表的数据查询出来，然后可以绑定到表单内的各个输入项中，但需要设置好这些输入项的数据绑定格式，否则无法绑定。设置方法：选中输入项，然后点击属性设置里的“数据绑定格式”，然后选择要绑定的字段即可。

各个输入项的数据绑定格式见下表：

| 组件编号            | 组件说明      | 数据绑定格式            |
|-----------------|-----------|-------------------|
| code            | 订单编码      | {code}            |
| operator_id     | 经办人       | {operator_id}     |
| order_date      | 下单日期      | {order_date}      |
| customer_id     | 客户编号(隐藏域) | {customer_id}     |
| customer_name   | 所属客户      | {customer_name}   |
| contacts        | 联系人       | {contacts}        |
| contact_number  | 联系电话      | {contact_number}  |
| order_amount    | 订单金额      | {order_amount}    |
| discount_amount | 优惠金额      | {discount_amount} |
| actual_amount   | 实收金额      | {actual_amount}   |
| comment         | 备注        | {comment}         |

### 5.9.6 设置可编辑表格数据查询参数

选中可编辑表格 dataSet1，点击属性里的数据查询设置，打开如下所示对话框：

在此，先选择销售订单明细表的数据模型：demo\_sm\_sale\_order\_detail，然后全选所有字段，然后添加产品信息有关的关联字段，如下表所示：

| 关联表             | 关联字段  | 字段别名  | 字段备注 | 关联条件                                    |
|-----------------|-------|-------|------|-----------------------------------------|
| demo_sm_product | name  | name  | 产品名称 | id=demo_sm_sale_order_detail.product_id |
| demo_sm_product | model | model | 规格型号 | id=demo_sm_sale_order_detail.product_id |

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型

请求参数

查询条件

分组和排序

最多查询 ☐ 条记录 ☐ 去掉重复记录 ☐ 不校验是否登录 ☐ 不检查SQL注入

数据模型路径：

选择...

添加自定义字段

添加关联字段

|                                     | 字段名称       | 字段备注 | 字段格式(SQL, 双击可编辑)                | 字段别名  | 数据字典(双击选择) |   |
|-------------------------------------|------------|------|---------------------------------|-------|------------|---|
| <input checked="" type="checkbox"/> | id         | 明细编号 |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | order_id   | 订单编号 |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | product_id | 产品编号 |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | price      | 单价   |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | quantity   | 数量   |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | unit       | 计量单位 |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | amount     | 金额   |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | comment    | 备注   |                                 |       |            | ⊖ |
| <input checked="" type="checkbox"/> | name       | 产品名称 | (select name from demo_sm_produ | name  |            | ⊖ |
| <input checked="" type="checkbox"/> | model      | 规格型号 | (select model from demo_sm_prod | model |            | ⊖ |

清空

确定

取消

然后点击“请求参数”选项卡，在此添加发送给后台的请求参数，设置界面如下：

数据查询设置

查询数据方式：☒ 数据模型 ☐ 数据表单 ☐ 后台服务 ☐ SQL语句 ☐ HTTP请求

数据模型

请求参数

查询条件

分组和排序

+

↓ 在此设置需要传递给后台的查询参数

| 参数名 | 参数类型   | 参数值 |         |
|-----|--------|-----|---------|
| id  | URL 参数 |     | ⊕ ⊖ ⊕ ⊖ |

清空

确定

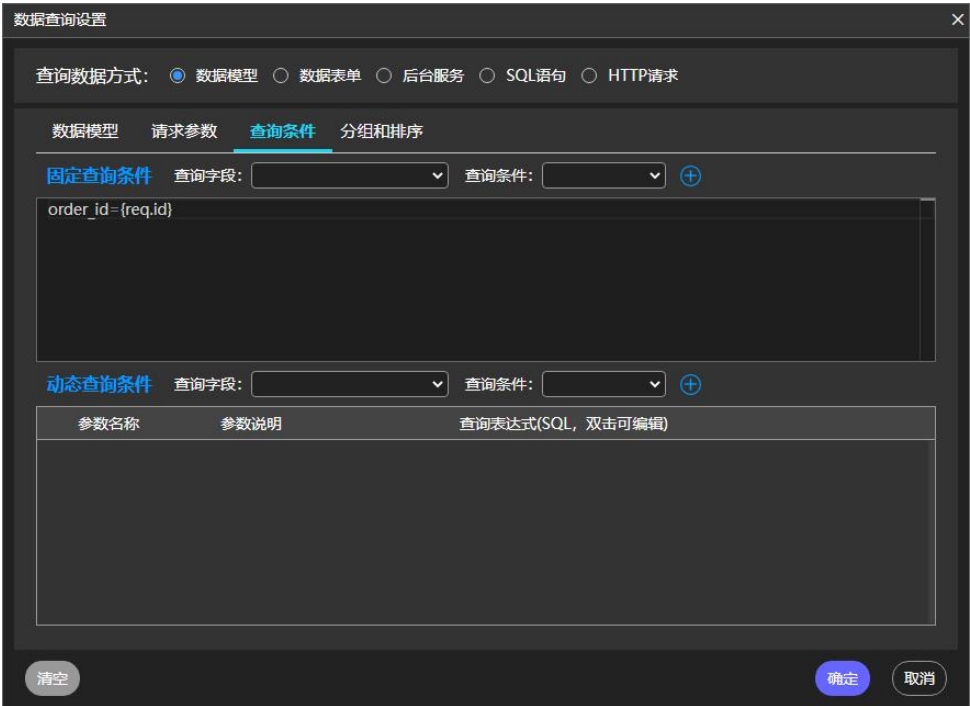
取消

然后点击“查询条件”选项卡，在此设置查询订单明细的查询条件，如下图所示：

北京任讯信息技术有限公司

435

www.taskbuilder.org



然后点确定，即可完成查询订单明细数据的设置。

### 5.9.7 修改表单数据提交设置

选中表单组件 form1，点击属性里的数据提交设置，打开如下所示对话框：



因为这个页面是从添加销售订单页面另存的，所以在这个对话框里，可以看到数据模型和字段都已经选择好了。在此，首先需要把提交类型修改为“修改”，然后取消选中以下字段：

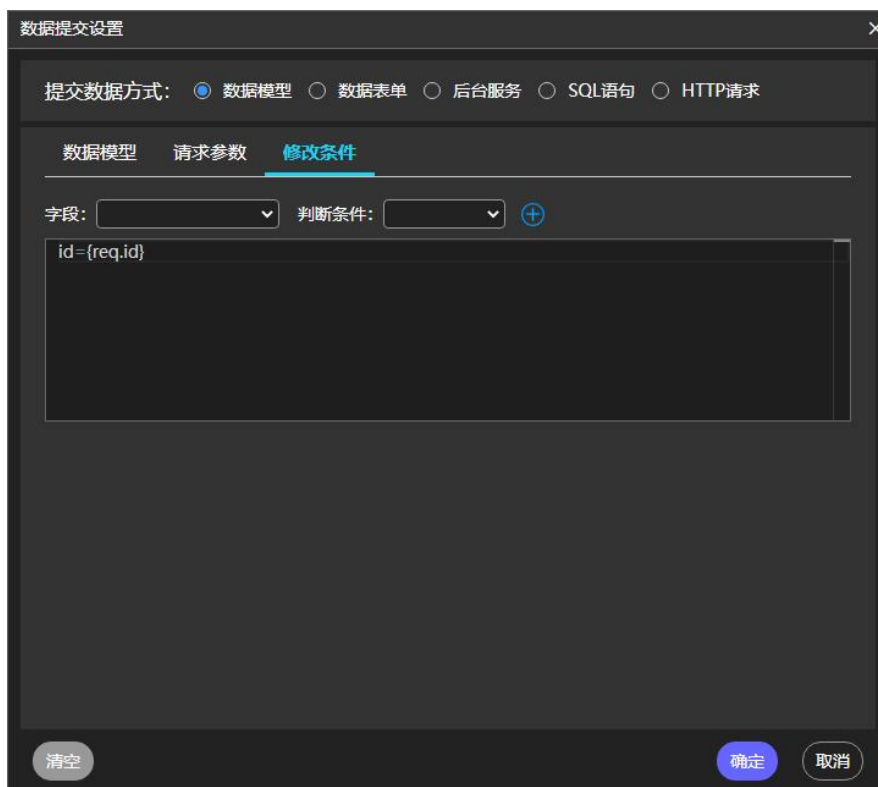
creator\_id、create\_time、status

因为这几个字段在修改销售订单时不需要设置值。

然后点击“请求参数”选项卡，在此设置需要发送给后台服务的请求参数，如下图所示：



然后点击“修改条件”选项卡，在此设置修改订单主表数据的修改条件，都设置好后，点确定按钮即可，修改条件设置界面如下图所示：



### 5.9.8 修改可编辑表格数据提交设置

选中可编辑表格 dataSet1，点击属性设置里的“数据提交设置”，打开如下所示对话框：



在此，先将提交类型修改为“修改”，然后将 order\_id 字段的值设置为 req.id，product\_id

字段的值设置为：#{product\_id}，然后点击“请求参数”选项卡设置请求参数，界面如下：

The screenshot shows the '数据提交设置' (Data Submission Settings) dialog box with the '请求参数' (Request Parameters) tab selected. The '提交数据方式' (Data Submission Method) is set to '数据模型' (Data Model). The '请求参数' tab contains a table for defining parameters to be passed to the backend.

| 参数名 | 参数类型 | 参数值 |
|-----|------|-----|
| id  | 默认   |     |

At the bottom of the dialog, there are buttons for '清空' (Clear), '确定' (Confirm), and '取消' (Cancel).

然后点击“修改条件”选项卡，在此设置修改订单明细数据的修改条件，如下图所示：

The screenshot shows the '数据提交设置' (Data Submission Settings) dialog box with the '修改条件' (Modify Conditions) tab selected. The '提交数据方式' (Data Submission Method) is still '数据模型' (Data Model). The '修改条件' tab allows setting conditions for modifying data.

Fields:  判断条件:  +

数据集批量修改选项: ☒ 先删除原来的数据，再添加本次提交的数据 不要在修改条件中设置主键字段

order\_id={req.id}

At the bottom of the dialog, there are buttons for '清空' (Clear), '确定' (Confirm), and '取消' (Cancel).



为了简化修改操作，建议选中“先删除原来的数据，再添加本次提交的数据”，否则需要挨条判断提交的每条明细数据是否已存在，存在就修改，不存在就添加，另外，还得判断原来的数据是否已删除，整个后台逻辑会很复杂。

都设置好后，点确定即可，然后保存页面，即可完成修改订单页面的设置。

## 5.9.9 预览测试修改销售订单页面

打开订单列表页面，然后点击工具栏上的预览按钮，打开预览窗口，然后点击某条订单数据所在行的修改按钮，即可打开该销售订单的修改界面进行修改测试，界面如下：

**修改销售单**

**基本信息**

订单编码: \* XS2024080019 经办人: \* 周瑾 下单日期: \* 2024-08-24

所属客户: \* 深圳某电气设备生产制造有限公司1 联系人: \* 李总2 联系电话: \* 136789054322

**订单明细**

选择产品

| 产品名称 | 规格型号        | 单价      | 数量 | 单位 | 金额       | 备注 |
|------|-------------|---------|----|----|----------|----|
| 电风扇  | 美的落地空气循环电风扇 | 189.00  | 10 | 包  | 1890.00  | 11 |
| 拖把   | 大卫免手洗平板拖把   | 731.90  | 20 | 吨  | 14638.00 | 22 |
| 电风扇  | 美的落地空气循环电风扇 | 1869.00 | 30 | 包  | 56070.00 | 33 |

**其他信息**

订单金额: \* 72598.00 优惠金额: 340.00 实收金额: \* 72258.00

备注: 手动调水岸东方

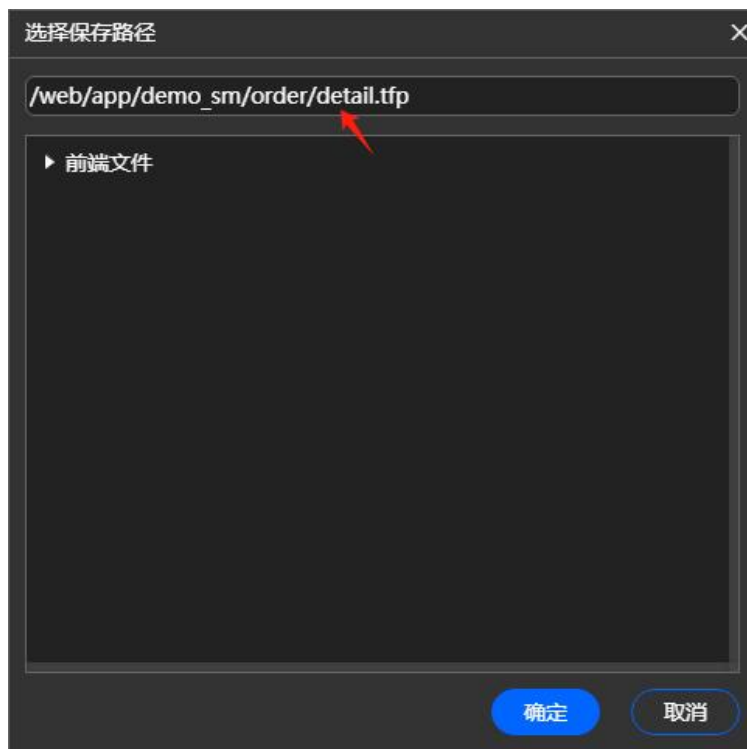
## 5.10 实现查看销售订单详情功能

### 5.10.1 创建查看销售订单详情页面

由于查看订单详情页面的内容与修改订单页面的内容基本一致，则可以通过将修改订单页

面另存来实现查看订单详情页面的创建，操作方法如下：

打开修改订单页面 `update.tfp`，点击工具栏上的另存为按钮，在弹出的菜单里选择另存，弹出如下所示对话框，在该对话框中将文件名修改为 `detail.tfp`，然后点确定即可完成创建。



### 5.10.2 去掉必填项提示

分别选中订单编码、经办人、下单日期、所属客户、联系人、联系电话、订单金额、实收金额这几个输入项的文本标签，在属性里，将附加内容设置为空，不显示星号后缀。

### 5.10.3 清除所属客户输入项的点击时事件

选中所属客户输入项，在事件设置里，删除“点击时”事件输入框里的内容。

### 5.10.4 将所有输入项设置为只读

因为查看订单详情页面只是用来查看订单数据，不需要修改，所以，需要将该页面里的所有输入项设置为只读，操作方法就是选中输入项，然后在属性设置里，把只读开关打开即可，如果输入项已经设置为只读，则不用管。

### 5.10.5 设置可编辑表格不能新增和修改明细数据

选中可编辑表格 dataSet1，在属性设置里，将以下选项关闭：

- 显示从弹窗选择添加
- 允许插入
- 允许更改位置
- 允许删除

然后点击数据列设置，将 price、quantity 和 comment 这几列设置为只读，然后点确定。

### 5.10.6 去掉输入项默认值

- (1) 选中经办人输入项 operator\_id，在属性设置里将“默认为当前用户”开关关上；
- (2) 选中下单日期输入项 order\_date，在属性设置里将“默认当前时间”开关关上；

### 5.10.7 增加审核信息

上面另存为的查看订单详情页面还需要增加审核有关的信息，具体操作如下：

#### 5.10.7.1 增加审核信息标题栏

为了能快速实现审核信息标题栏的添加，可以采取复制粘贴页面里已有的标题栏的形式来实现，操作方法如下：

- (1) 选中其他信息文本标签所在的弹性面板组件 flexPanel3，然后点工具栏上的复制按钮 ；
- (2) 选中表单组件 form1；
- (3) 点击工具栏上的粘贴按钮 ，将上面复制的内容粘贴到表单组件底部；
- (4) 选中新粘贴的标题栏里的其他信息文本标签，将其组件文本修改为：审核信息；

这样就完成了审核信息标题栏的设置。

### 5.10.7.2 增加审核信息输入项

和上面一样，审核信息输入项也可以通过复制粘贴来快速创建，操作方法如下：

- (1) 选中订单编码、经办人和下单日期这三个输入项所在行的弹性栅格组件 flexBox\_row\_1，然后点击工具栏上的复制按钮；
  - (2) 选中表单组件 form1；
  - (3) 点击工具栏上的粘贴按钮，将上面复制的内容粘贴到表单组件底部；
  - (4) 选中新粘贴的订单编码文本标签，在属性设置里将组件文本修改为“审核意见：”；
  - (5) 选中新粘贴的订单编码输入项，在属性设置里将数据绑定格式修改为：{audit\_opinion}；
  - (6) 选中新粘贴的经办人文本文本标签，在属性设置里将组件文本修改为“审核人：”；
  - (7) 选中新粘贴的经办人输入项，在属性设置里将数据绑定格式修改为：{auditor\_id}；
  - (8) 选中新粘贴的下单日期文本标签，在属性设置里将组件文本修改为“审核日期：”；
  - (9) 选中新粘贴的下单日期输入项，在属性设置里将数据绑定格式修改为：{audit\_time}；
- 这些都设置好后，就可以在订单列表里打开这个页面查看订单详情了。

### 5.10.8 销售订单列表增加查看订单详情功能

在订单列表页面 index.tfp 里，选中数据表格组件 grid1，在属性设置里点击数据列设置，在弹出的对话框中，在左侧选中订单编码列，然后在右边基本信息的“点击后执行的脚步”输入框中输入以下脚本：

```
tfp.openDrawer('订单详情', 'detail.tfp?id={id}')
```

然后点确定，然后保存订单列表页面。

### 5.10.9 预览测试查看订单详情页面

预览订单列表页面，然后点击某个订单的编码，即可打开该订单的详情页面，如下图所示：

刷新

添加

订单编码:  经办人:

| 订单编码         | 经办人 | 下单日期       | 深圳某 |
|--------------|-----|------------|-----|
| XS2024080028 | 周瑾  | 2024-08-18 | 深圳某 |
| XS2024080019 | 孙研  | 2024-08-24 | 深圳某 |

订单详情

关闭

基本信息

订单编码: XS2024080019 经办人: 孙研 下单日期: 2024-08-24

所属客户: 深圳某电气设备生产制造有限公司1 联系人: 李总2 联系电话: 136789054322

订单明细

| 产品名称 | 规格型号        | 单价      | 数量 | 单位 | 金额       | 备注 |
|------|-------------|---------|----|----|----------|----|
| 电风扇  | 美的落地空气循环电风扇 | 189.00  | 10 | 包  | 1890.00  | 11 |
| 拖把   | 大卫免手洗平板拖把   | 731.90  | 20 | 吨  | 14638.00 | 22 |
| 电风扇  | 美的落地空气循环电风扇 | 1869.00 | 30 | 包  | 56070.00 | 33 |

其他信息

订单金额: 72598.00 优惠金额: 340.00 实收金额: 72258.00

备注: 手动调

审核信息

审核意见: 审核人: 审核日期:

可以看到，因为还没有实现订单审核功能，所以该订单的审核信息时空的。

## 5.11 实现销售订单审核功能

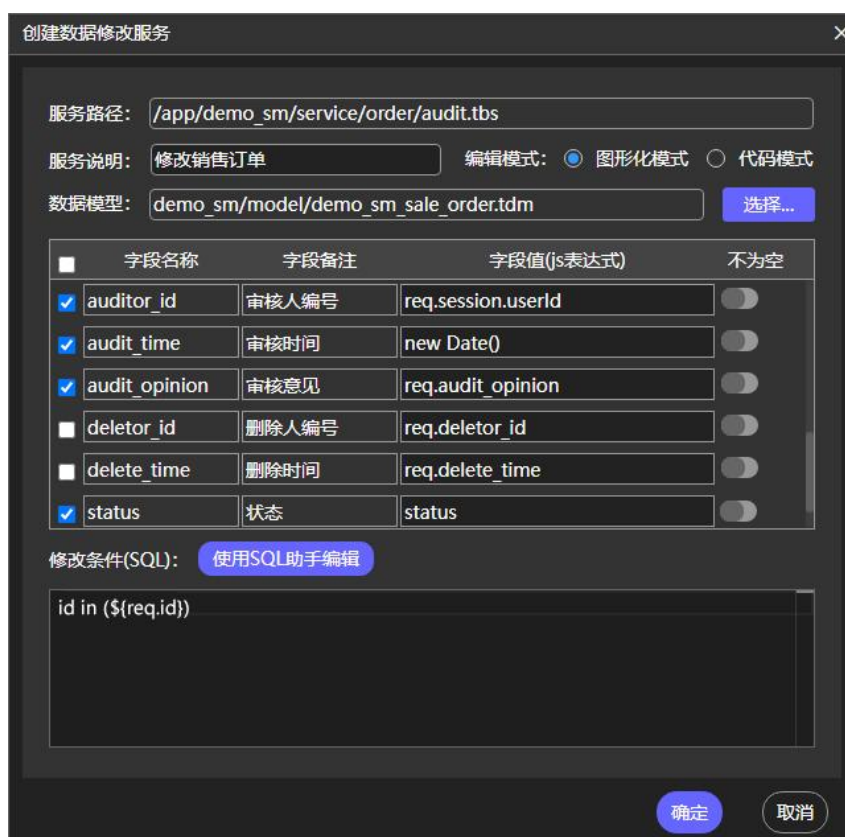
接下来我们实现订单审核的功能，这里实现的审核功能很简单，就是由一个专人操作，并简单地记录一下审核人、审核时间和审核意见，如果单据需要经过多个人审核，建议采用 TaskFlow 工作流引擎来实现。

### 5.11.1 创建订单审核后台服务

点击销售管理示例项目的后台服务里的 order 目录右侧的加号，打开如下图所示的“后台资源创建向导”，在里面选择“数据修改服务”，资源名称设置为：audit。



然后点下一步，打开如下所示的“创建数据修改服务”的对话框：





在该对话框中，先选则销售订单数据模型：demo\_sm\_sale\_order.tdm，然后按照下表所示选中并设置以下字段的值：

| 字段名称          | 字段备注  | 字段值                                                   |
|---------------|-------|-------------------------------------------------------|
| auditor_id    | 审核人编号 | req.session.userId（当前用户编号）                            |
| audit_time    | 审核时间  | new Date()（当前日期时间）                                    |
| audit_opinion | 审核意见  | req.audit_opinion                                     |
| status        | 单据状态  | status（后台变量，根据审核意见赋不同的值）<br>要把不为空开关关闭，因为这个字段的值是在后台设置的 |

修改条件设置为：`id in (${req.id})`，表示将前端传递的 id 参数作为字符串直接拼接接到 SQL 修改条件中，这样就可以支持以英文逗号分隔的多个 id 值。

都设置好后，点击确定，即可创建该后台服务，然后按照下图所示修改完善该后台服务：



从上图可知，在修改订单的审核信息之前，声明了一个名为 status 的变量，默认值为 1，表示审核通过，然后判断前端页面传递的审核意见参数（audit\_opinion）的值如果等于“不同意”，则将 status 变量的值设置为 2，表示不同意，这样设置好后，在下面的数据修改语句里，就可以根据审核意见将销售订单的状态设置为不同的值。

另外，点击后台服务设计器底部的“基本信息”，在请求参数列表里，将 id 参数的类型改成字符串，因为页面可能会传以逗号分隔的多个 id 值，以便实现批量审核。

### 5.11.2 创建订单审核对话框页面

点击销售管理示例项目的前端文件里的 order 目录右侧的加号，打开如下图所示的“前端资源创建向导”，在里面选择“数据添加 TFP”，资源名称设置为：audit。



设置好后，点击下一步按钮，打开如下图所示的“创建数据添加 TFP”对话框，在该对话框中，先选择销售订单数据模型：demo\_sm\_sale\_order.tdm，然后设置以下参数：

- **页面标题**：设置为订单审核；
- **页面类型**：选择对话框；
- **窗口大小**：宽度设置为 240，高度设置为 180；
- **每行显示输入项数量**：设置为 1；
- **数据添加服务**：选择使用上面创建的订单审核服务：demo\_sm/service/order/audit.tbs；

| 字段                                                | 输入项名称 | 输入项类型 | 列数 | 必填                                  | 可选值    | 数据字典(双击选择) | 默认值 |
|---------------------------------------------------|-------|-------|----|-------------------------------------|--------|------------|-----|
| <input type="checkbox"/> creator_id               | 创建人编号 | 单行输入框 | 1  | <input checked="" type="checkbox"/> |        |            |     |
| <input type="checkbox"/> create_time              | 创建时间  | 单行输入框 | 1  | <input checked="" type="checkbox"/> |        |            |     |
| <input type="checkbox"/> auditor_id               | 审核人编号 | 单行输入框 | 1  | <input type="checkbox"/>            |        |            |     |
| <input type="checkbox"/> audit_time               | 审核时间  | 单行输入框 | 1  | <input type="checkbox"/>            |        |            |     |
| <input checked="" type="checkbox"/> audit_opinion | 审核意见  | 单选框   | 1  | <input type="checkbox"/>            | 同意,不同意 |            |     |
| <input type="checkbox"/> deleter_id               | 删除人编号 | 单行输入框 | 1  | <input type="checkbox"/>            |        |            |     |
| <input type="checkbox"/> delete_time              | 删除时间  | 单行输入框 | 1  | <input type="checkbox"/>            |        |            |     |
| <input type="checkbox"/> status                   | 状态    | 单行输入框 | 1  | <input checked="" type="checkbox"/> |        |            |     |
| <input type="checkbox"/> comment                  | 备注    | 单行输入框 | 1  | <input type="checkbox"/>            |        |            |     |

然后在下面的“前端页面设置字段”中，选中 audit\_opinion 字段，输入项类型设置为单选框，可选值设置为“同意,不同意”，然后点确定即可完成订单审核页面的创建。

### 5.11.3 设置订单审核页面表单组件的请求参数

在上面创建的订单审核页面里，选中表单组件 form1，然后点击数据提交设置，在弹出的对话框里点击“请求参数”选项卡，在里面添加一个名为 id 的参数，参数类型选择“URL 参数”，表示在提交审核数据时，会将该页面的 URL 里的 id 参数传给订单审核的后台服务，而这个 id 参数是在点击列表页面里的审核按钮后传过来的，后面会讲解如何设置这个按钮。

### 5.11.4 订单列表页面增加审核功能

接下来需要修改订单列表页面，增加订单审核有关的功能。

#### 5.11.4.1 订单列表显示选择框

在订单列表页面里，选中数据表格组件 grid1，打开属性设置里的“显示选择框”开关。

### 5.11.4.2 增加和设置审核按钮

从组件库的只读组件中拖拽一个按钮到页面上部查询按钮的后面，并设置以下属性：

- **按钮文本**：设置为审核；
- **按钮类型**：设置为危险按钮；
- **显示图标**：打开该开关；
- **图标**：选中确定图标；

然后在样式设置的边距和边框中，将外边距的左边距设置为 15。

然后在该按钮的事件设置里，双击“点击时”事件输入框，进入事件脚本编辑器，在此按下图所示拖拽并设置相关语句：



上图所示语句的含义解释如下：

(1) 先调用数据表格组件 grid1 的获得选中行的主键值方法，并把结果（选中行的订单的 id，多个 id 以逗号分隔）保存到名为 keys 的变量中，这个变量名可以在这个语句的参数设置对话框中设置；

(2) 判断这个 keys 变量是否为空，如果为空，表示没有选中任何订单，则弹出提示框提示，并退出该事件函数，不再执行后面的语句；

(3) 如果选中了一行或多行，则打开订单审核对话框，并把选中的主键值传给该页面；在该语句的参数设置中，要将对话框宽度设置为 240px，高度设置为 180px。

至此，就完成了订单审核的功能。

### 5.11.5 预览测试订单审核功能

打开订单列表页面，点击预览按钮，在打开的预览窗口中，选择要审核的订单，然后点击审核按钮，选中审核意见，即可完成审核，操作界面如下图所示：



由于前面创建订单列表时，显示的是所有状态的订单，所以审核完后，订单列表里的数据并不会有什么变化，下面会重构订单列表页面，按状态分多个选项页显示。

## 5.12 实现订单列表按状态显示

为了方便对不同状态的订单进行查看和管理，下面准备重构订单列表页面，通过选项卡组件将不同状态的订单分别显示在不同的选项页里，具体操作方法如下。

### 5.12.1 重命名列表页面

因为列表页面需要做比较大的改动，如果直接在原来的列表页面上修改，操作比较繁琐，但原来的内容还可以作为参考，不能直接删除，所以，需要把原来的列表页面重命名为其他文件名，操作方法如下：

在销售管理示例项目的前端文件的 order 目录里，在 index.tfp 页面上点击鼠标右键，然后选择“重命名”，会弹出如下对话框：

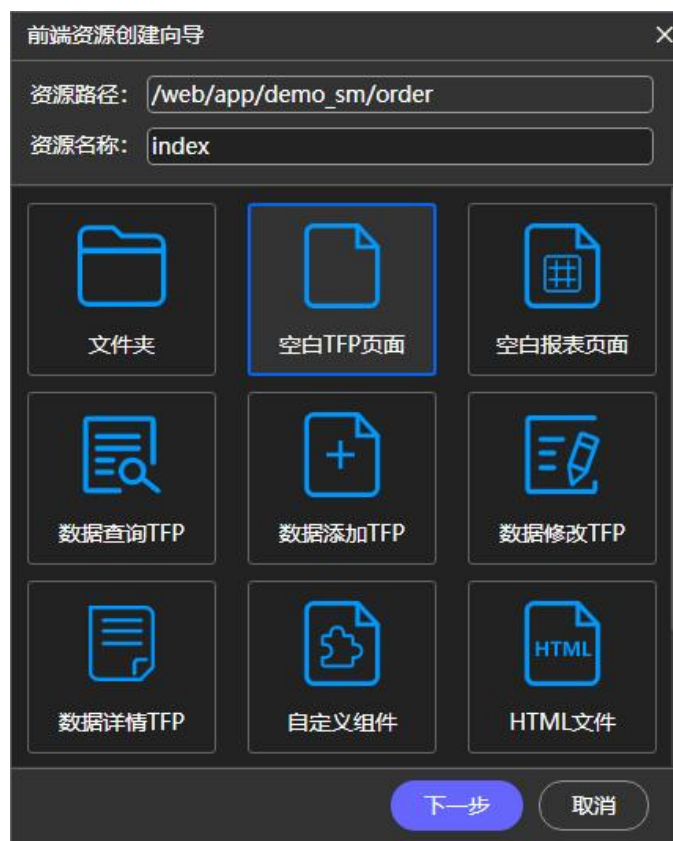


在此，将文件名修改为 index\_bak，表示该文件为备份文件，非正式页面。

### 5.12.2 重新创建空白的列表页面

在销售管理示例项目的前端文件里，点击 order 目录右侧的加号，打开下图所示的“前端资源创建向导”对话框，选中“空白 TFP 页面”，资源名称设置为：index。





点下一步会显示如下图所示的“创建空白 TFP 页面”，在此将页面标题设置为：销售订单管理，并选中“创建页面背景面板”。



然后点确定，即可完成该页面的创建。

### 5.12.3 在新列表页面上添加和设置选项卡组件

在新创建的空白页面里，从组件库的容器组件中，拖拽一个选项卡组件到页面的背景面板组件中，然后在该组件的属性设置中，点击“选项页集合”，在弹出的对话框中参考下图设置三个选项页：



然后点击确定，即可为该组件设置三个选项页，界面如下图：



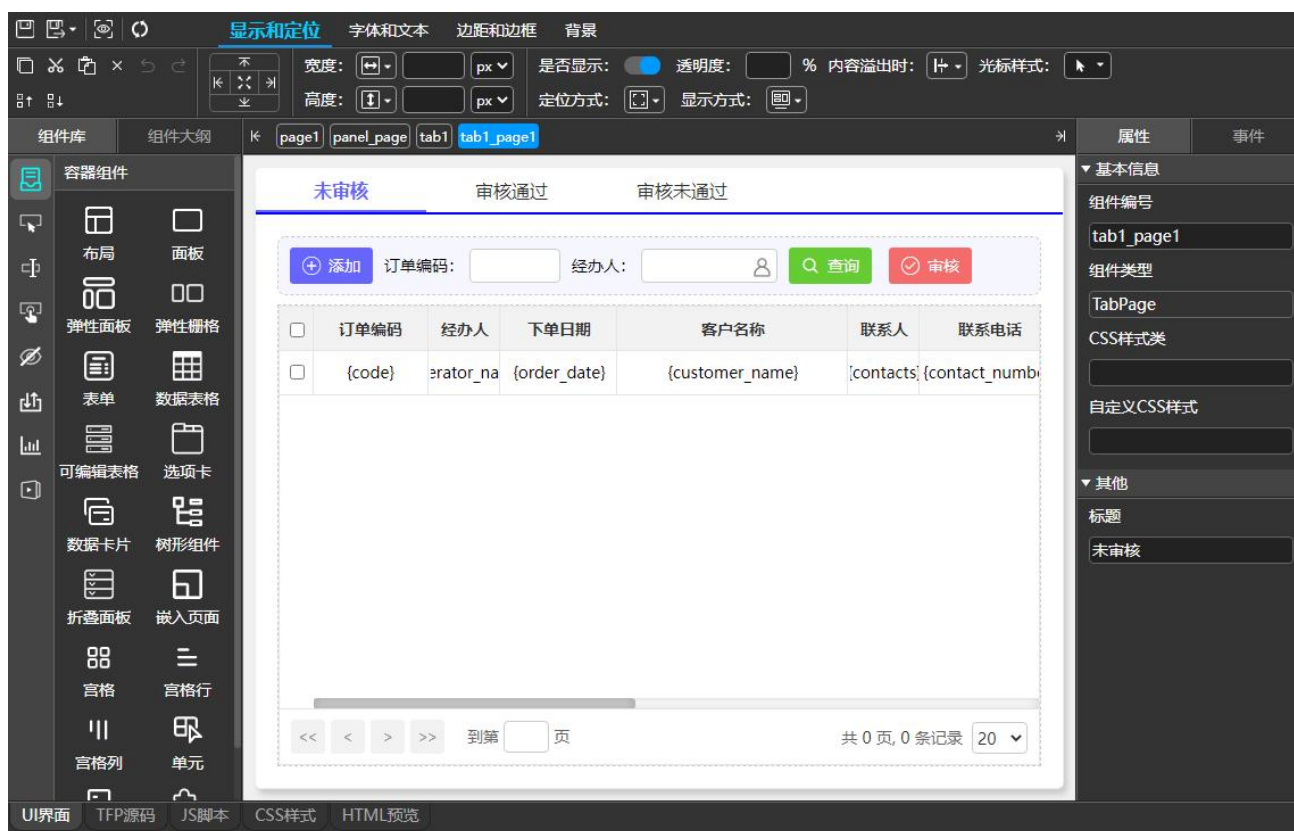
## 5.12.4 创建和设置未审核订单列表

我们先实现“未审核”选项页里组件的复制粘贴，操作步骤如下：

(1) 打开原来的列表页面 index\_bak.tfp，选中页面上部用来放置查询条件输入项的面板组件 panelQuery，然后点击工具栏上的复制按钮：；

(2) 打开新建的列表页面 index.tfp，选中第一个选项页 tab1\_page1，然后点击工具栏上的粘贴按钮：，即可将这个组件及其子组件都复制到新的列表页面里；

按照上述操作方法，把原来列表页面里的数据表格组件 grid1 也复制粘贴到新列表页面的第一个选项页里，操作完成后的界面如下图所示：

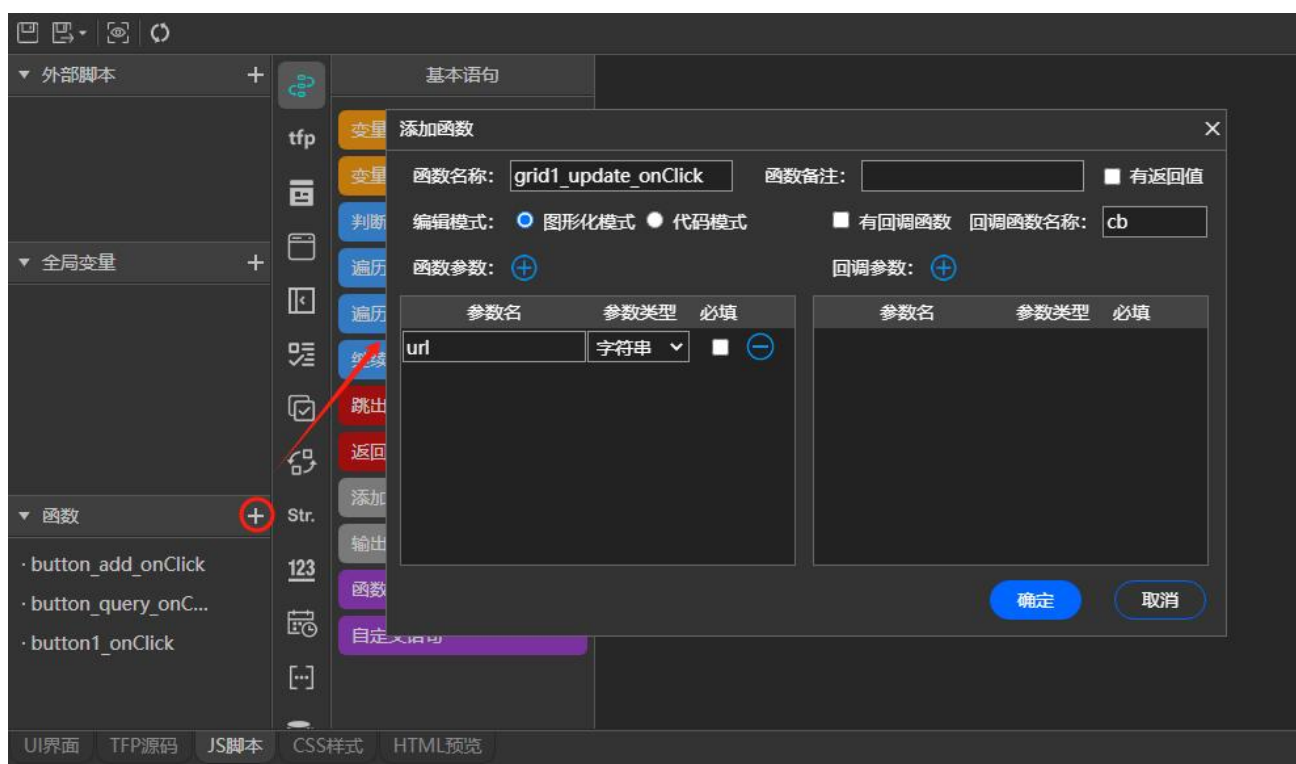


然后选中数据表格组件 grid1，点击属性设置里的数据查询设置，然后点击“查询条件”，将固定查询条件设置为：status=0，表示这个数据表格仅查询未审核的订单。

另外，复制粘贴组件时，会自动将组件及其子组件的事件处理函数一同复制到新页面里，例如添加、查询和审核这几个按钮的点击事件的处理函数，在复制粘贴查询面板组件时，就自

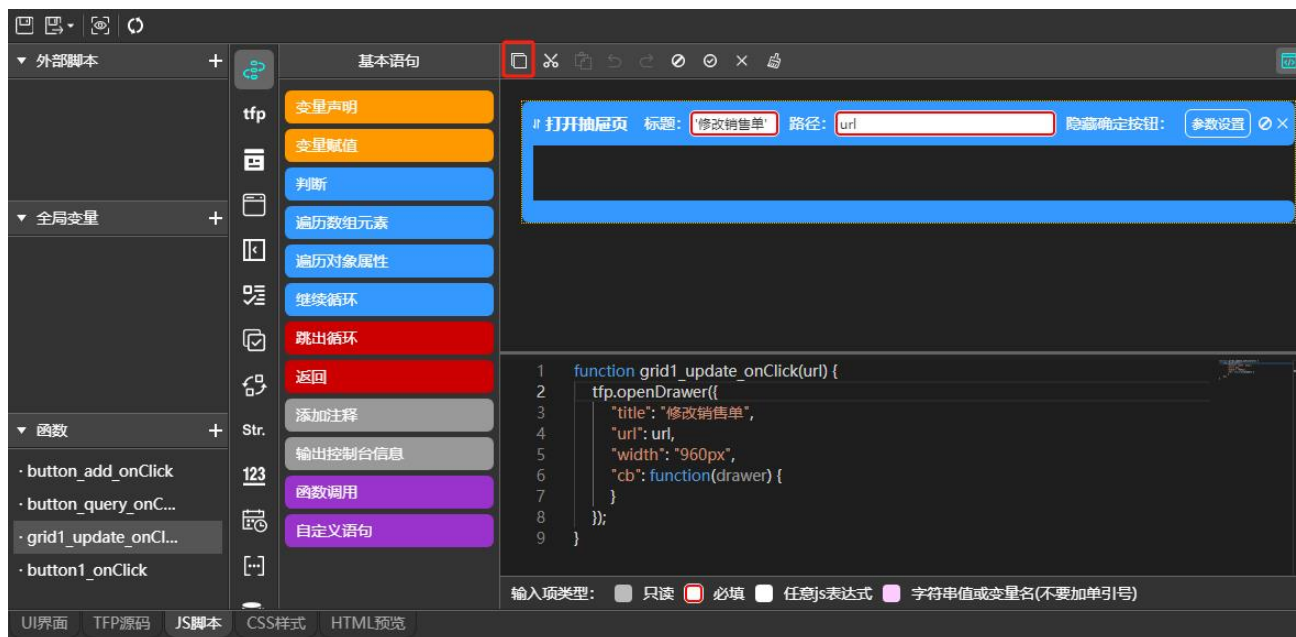
动复制到新的列表页面里了，但还有一个脚本函数需要单独复制，那就是点击订单列表里的修改按钮后执行的函数 `grid1_update_onClick`，这个函数是向导自动生成的一个自定义函数，不是组件的事件处理函数，所以无法在复制粘贴组件时自动带过来，需要手动创建和复制里面的语句，否则，在列表里点击修改按钮时就会报错，操作方法如下：

(1) 在新建的订单列表页面里，点击底部的“JS 脚本”，然后再点击左侧函数列表标题栏右侧的加号，打开如下图所示的添加函数对话框：



在此，将函数名称设置为：`grid1_update_onClick`，然后添加一个名为 `url` 的函数参数，然后点确定。

(2) 在原来的订单列表页面里，点击底部的“JS 脚本”，然后再点击左侧函数列表里的 `grid1_update_onClick` 函数，然后点击右侧可视化脚本设计器里的“打开抽屉页”语句，选中该语句（出现黄色的虚线框），然后点击工具栏上的复制按钮：；



(3) 然后回到新建的订单列表页面，在 JS 脚本编辑器里，选中 grid1\_update\_onClick 函数，然后点击工具栏上的粘贴按钮：，即可将原来列表页面里的这个函数里的语句粘贴到新的列表页面里；如果新页面里这个粘贴按钮显示不可用，应该是复制之前，这个界面就没动，需要在函数列表里点一下其他函数，然后再点 grid1\_update\_onClick 函数，就能看到可以用了；

然后保存新建的这个列表页面，点击工具栏的预览图标，即可查看运行效果，如下图所示：



## 5.12.4 创建和设置审核通过订单列表

在新的列表页面里，点击“审核通过”选项卡，下面会显示其对应的选项页，目前是空白的，我们可以参考上面的操作，将原来的订单列表页面里的顶部面板组件和数据表格组件复制粘贴到这个空白的选项页里，具体操作就不再复述。

复制粘贴完后，需要做以下完善：

### 5.12.4.1 删除添加和审核按钮

审核通过页面不需要“添加”按钮，可以选中该按钮，然后点击工具栏上的删除图标即可删除该按钮。另外，已审核的订单不能再审核，所以审核按钮也需要删除。

然后点击底部的“JS 脚本”，将添加和审核按钮对应的事件处理函数 `button2_onClick` 和 `button4_onClick` 也删除掉。

### 5.12.4.2 去掉选择框

选中“审核通过”选项页里的数据表格组件，在属性里把显示选择框开关关闭。

### 5.12.4.3 不显示修改和删除按钮

选中“审核通过”选项页里的数据表格组件，点击属性设置里的数据列设置，把修改和删除这两列删除。

### 5.12.4.4 修改查询条件

因为未审核选项页里已经有编号为 `code` 和 `operator_id` 的输入项，所以为了避免组件编号重复，从原列表页面复制查询条件面板到“审核通过”选项页里时，订单编码和经办人这两个输入项的编号系统自动设置成了 `text1` 和 `user1`，为了便于理解，需要分别改成 `code2` 和 `operator_id2`，表示是第 2 个选项页面里的订单编码和经办人这两个输入项。

然后选中“审核通过”选项页里的数据表格组件，点击属性设置里的数据查询设置，在弹出的对话框中，点击“请求参数”选项卡，将里面已有的两个请求参数的值改成 `code2` 和



operator\_id2，如下图所示：

| 数据模型 请求参数 查询条件 分组和排序 |      |              |
|----------------------|------|--------------|
| 参数名                  | 参数类型 | 参数值          |
| code                 | 组件值  | code2        |
| operator_id          | 组件值  | operator_id2 |

然后点击“查询条件”选项卡，将固定查询条件设置为：status=1，表示仅查询审核通过的订单，设置完毕后点击确定。

最后，选中“查询”按钮，在事件设置里，双击“点击时”事件输入框，打开下图所示事件函数编辑器，将里面的grid1改为gridView1，方法还是选择“重新加载数据”。



都设置好后，保存页面，预览，点击“审核通过”选项卡，即可看到如下图所示界面：

刷新

本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

未审核

审核通过

审核未通过

+ 添加

订单编码:

经办人:

人

Q 查询

| 订单编码         | 经办人 | 下单日期       | 客户名称             | 联系人 | 联系电话         | 订单金额      | 优惠金额   | 实收金额      |
|--------------|-----|------------|------------------|-----|--------------|-----------|--------|-----------|
| XS2024080028 | 周瑾  | 2024-08-18 | 深圳某电气设备生产制造有限公司1 | 李总1 | 136789054321 | 3,840.00  | 50.00  | 3,790.00  |
| XS2024080019 | 孙研  | 2024-08-24 | 深圳某电气设备生产制造有限公司1 | 李总2 | 136789054322 | 72,598.00 | 340.00 | 72,258.00 |

<<

<

1

>

>>

到第

1

页

共 1 页, 2 条记录 20

### 5.12.5 创建和设置审核未通过订单列表

审核未通过订单列表与审核通过订单列表的界面基本一致，所以可以直接将“审核通过”选项页里的查询面板和数据表格组件直接复制到“审核未通过”选项页里，实现该页面功能的快速创建，具体操作参考前面的描述。

复制粘贴好后，还需要修改查询条件，操作方法如下：

将顶部查询面板里的订单编码和经办人这两个输入项的编号分别改成 `code3` 和 `operator_id3`，表示是第 3 个选项页面里的订单编码和经办人。

然后选中该选项页里的数据表格组件，点击属性设置里的数据查询设置，在弹出的对话框中，点击“请求参数”选项卡，将里面已有的两个请求参数的值改成 `code3` 和 `operator_id3`。

最后，选中“查询”按钮，在事件设置里，双击“点击时”事件输入框，打开下图所示事件函数编辑器，将里面的 `grid1` 改为 `gridView2`，方法还是选择“重新加载数据”。

设置好后，点保存，再点预览，即可查看该页面的运行效果。

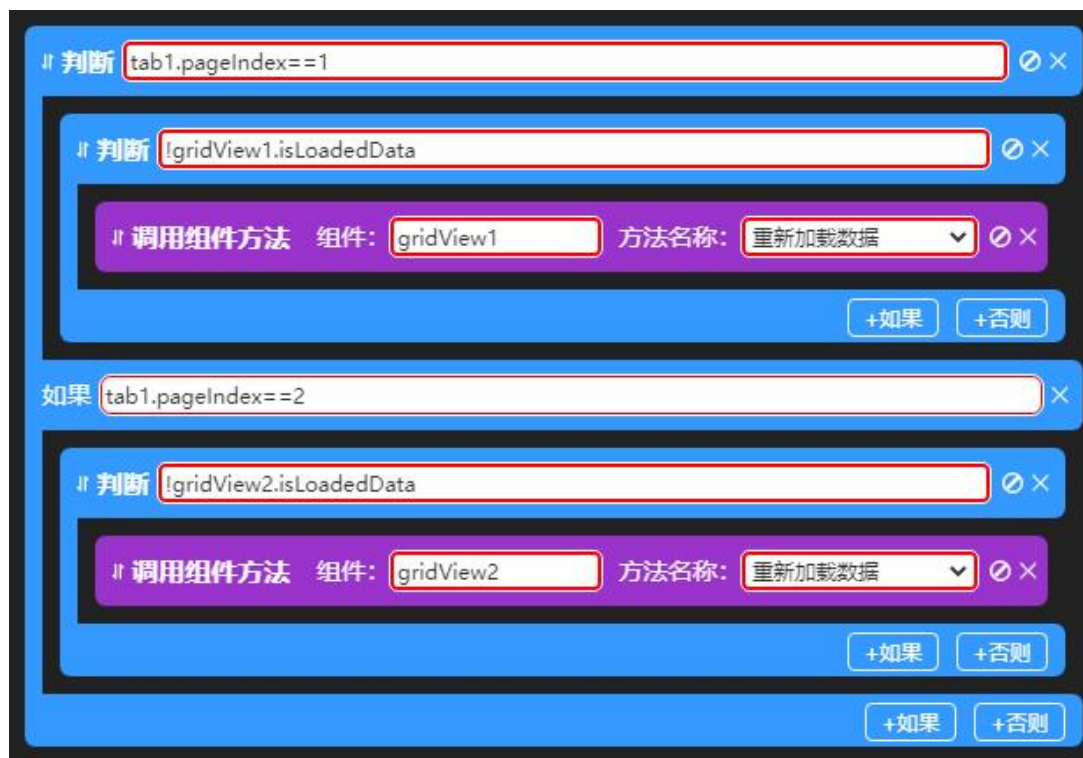
## 5.13 设置列表数据懒加载

上面创建完的订单列表页面里，放置了三个数据表格组件，在打开页面时，这三个数据表格组件都会请求后台查询数据，数据多了后，打开这个页面会比较慢，如果同时很多人打开，会给服务器造成较大压力，但实际上很多时候用户每次只会查看某一种状态的订单列表，很少要全部查看，所以，为了提升系统性能，我们可以对该页面做些优化，实现数据的“懒加载”，操作步骤如下：

(1) 分别选中“审核通过”和“审核未通过”这两个选项页里的数据表格组件，在属性设置里，将“不自动加载数据”开关打开；

(2) 选中选项卡组件 `tab1`，在事件设置里，双击“选中页变化时”事件输入框，进入事件函

数编辑界面，按照下图所示添加相关语句：



上图所示语句的含义解释如下：

(1) 如果当前选项卡组件显示的是第二个选项页（`tab1.pageIndex==1`），则判断第二个选项页里的数据表格组件是否还没有加载数据（`!gridView1.isLoadedData`），如果没有，则加载该表格的数据；

(2) 如果当前选项卡组件显示的是第三个选项页（`tab1.pageIndex==1`），则判断第三个选项页里的数据表格组件是否还没有加载数据（`!gridView2.isLoadedData`），如果没有，则加载该表格的数据；

这样设置以后，再打开这个页面时，默认就只会查询“未审核”的订单信息，如果用户点击了“审核通过”或“审核未通过”选项卡，才会查询这两个选项页里的列表数据，这样就减少了不必要的数据查询，减轻了服务器压力，是很多成熟的信息系统普遍采用的一种数据加载方式。

## 5.14 发布销售订单管理应用

整个销售订单管理模块开发完成后，就可以在后台管理系统的系统维护的应用管理中，添加对应的应用信息，相关信息设置如下：

- **应用代码：**设置为 demo\_sm\_order；
- **应用名称：**设置为销售订单；
- **应用分类：**选前面创建的“示例项目”这个类；
- **支持终端：**选择 Web 端；
- **web 端首页 URL 路径：** /app/demo\_sm/order/index.tfp；

添加完应用后，还可以添加该应用的操作信息和访问权限，具体如何设置请参考前面客户管理模块里的相关介绍，这里就不再复述。

至此，销售订单管理模块的功能就开发完毕了。

## 6、销售统计模块开发

### 6.1 本节知识点

- (1) 创建视图类型的数据模型；
- (2) 数据统计后台服务的设计；
- (3) 数据统计报表页面的实现；
- (4) 分组统计的实现
- (5) 行转列统计的实现
- (6) 常用统计图组件的使用

## 6.2 创建数据统计所需的视图

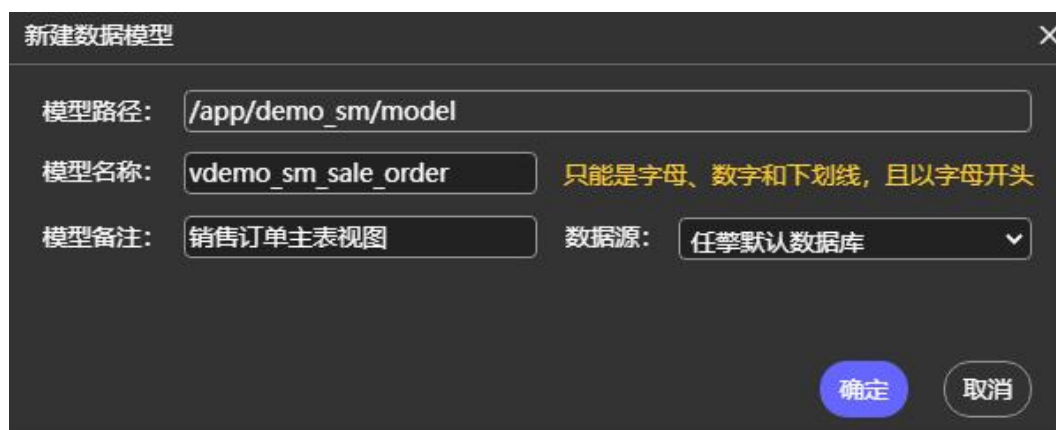
各个统计报表中需要关联查询多个表的数据，为了方便后续操作，需要先创建好所需的视图，包括销售订单主表视图和销售订单明细视图，具体创建方法如下：

### 6.2.1 创建销售订单主表视图

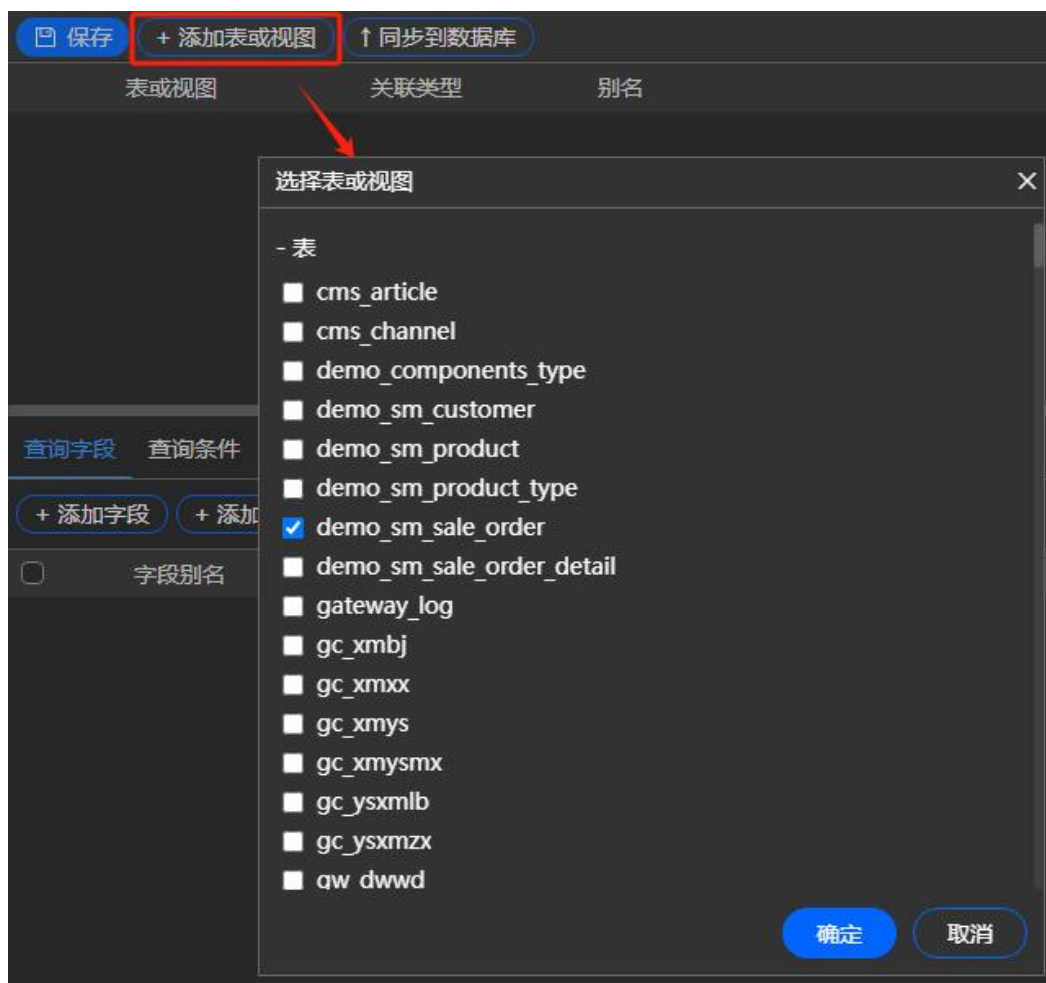
在销售管理示例项目的数据模型中，点击右侧的加号，打开如下所示对话框：



在此选中“新建视图”，然后点击下一步，打开如下对话框：



在此，将数据模型名称设置为：vdemo\_sm\_sale\_order，模型备注设置为：销售订单主表视图，然后点确定，打开如下所示的视图设计界面：



先点击工具上的“添加表和视图”按钮，选择 demo\_sm\_sale\_order 表，然后点确定，系统会自动将该表作为视图的主表，然后继续点击工具上的“添加表和视图”按钮，选中 demo\_sm\_customer 和 sys\_user 这两个表，点确定。

然后设置关联表的关联条件为：

demo\_sm\_customer.id = demo\_sm\_sale\_order.customer\_id

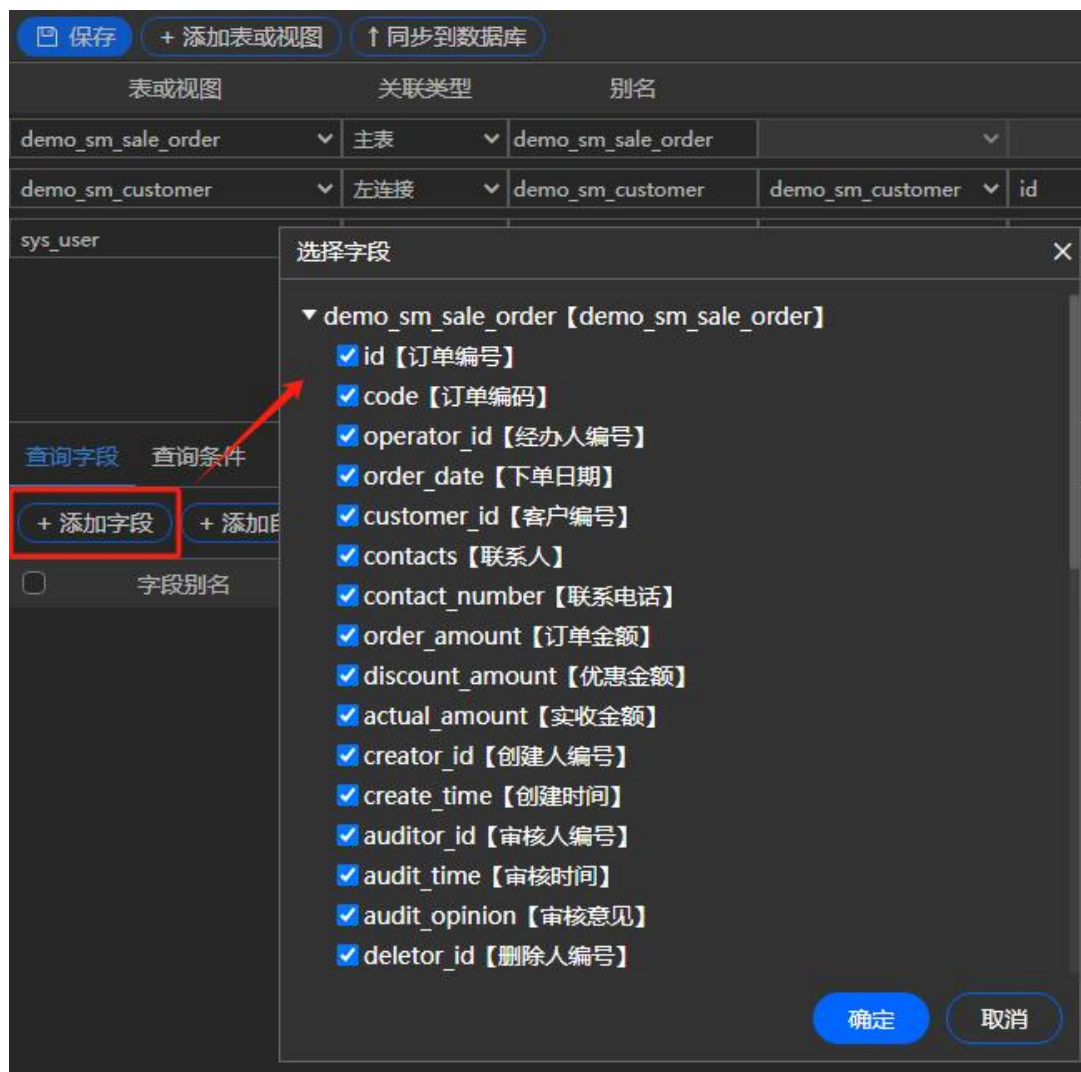
sys\_user.id = demo\_sm\_sale\_order.operator\_id

设置好后的界面如下图所示：

| 表或视图               | 关联类型 | 别名                 | 关联条件                                                 |
|--------------------|------|--------------------|------------------------------------------------------|
| demo_sm_sale_order | 主表   | demo_sm_sale_order |                                                      |
| demo_sm_customer   | 左连接  | demo_sm_customer   | demo_sm_sale_order.customer_id = demo_sm_customer.id |
| sys_user           | 左连接  | sys_user           | demo_sm_sale_order.operator_id = sys_user.id         |

然后在下面的查询字段选项页里，点击添加字段按钮，选中 demo\_sm\_sale\_order 表中的所有字段，demo\_sm\_customer 和 sys\_user 表中的 name 字段，然后点确定：





然后在字段列表里, 把 demo\_sm\_customer 表的 name 字段的别名修改为 customer\_name, 字段备注设置为客户名称, 把 sys\_user 表的 name 字段的别名修改为 operator\_name, 字段备注设置为经办人姓名, 如下图所示:

查询字段 查询条件 分组条件 过滤条件

+ 添加字段 + 添加自定义字段 - 删除字段 ☐ 排除重复

| <input type="checkbox"/> | 字段别名          | 字段格式               | 字段备注          |       |
|--------------------------|---------------|--------------------|---------------|-------|
| <input type="checkbox"/> | creator_id    | demo_sm_sale_order | creator_id    | 创建人编号 |
| <input type="checkbox"/> | create_time   | demo_sm_sale_order | create_time   | 创建时间  |
| <input type="checkbox"/> | auditor_id    | demo_sm_sale_order | auditor_id    | 审核人编号 |
| <input type="checkbox"/> | audit_time    | demo_sm_sale_order | audit_time    | 审核时间  |
| <input type="checkbox"/> | audit_opinion | demo_sm_sale_order | audit_opinion | 审核意见  |
| <input type="checkbox"/> | deletor_id    | demo_sm_sale_order | deletor_id    | 删除人编号 |
| <input type="checkbox"/> | delete_time   | demo_sm_sale_order | delete_time   | 删除时间  |
| <input type="checkbox"/> | status        | demo_sm_sale_order | status        | 状态    |
| <input type="checkbox"/> | comment       | demo_sm_sale_order | comment       | 备注    |
| <input type="checkbox"/> | customer_name | demo_sm_customer   | name          | 客户名称  |
| <input type="checkbox"/> | operator_name | sys_user           | name          | 经办人姓名 |

都设置好后，点击工具栏上的保存按钮，会弹出提示“是否同步到数据库”，点击确定，即可实现该视图的创建，并同步到数据库。

## 6.2.2 创建销售订单明细视图

在销售管理示例项目的数据模型中，点击右侧的加号，在打开的对话框中选中“新建视图”，然后点击下一步，打开如下对话框：

新建数据模型

模型路径: /app/demo\_sm/model

模型名称: vdemo\_sm\_sale\_order\_detail 只能是字母、数字和下划线，且以字母开头

模型备注: 销售订单明细视图 数据源: 任擎默认数据库

确定 取消

在此，将数据模型名称设置为：vdemo\_sm\_sale\_order，模型备注设置为：销售订单明细视图，然后点确定，打开视图设计界面，先添加 demo\_sm\_sale\_order\_detail 作为主表，然后再添加 demo\_sm\_sale\_order、demo\_sm\_product 和 demo\_sm\_product\_type 作为关联表，关联

北京任讯信息技术有限公司 465 www.taskbuilder.org

条件为：

`demo_sm_sale_order.id = demo_sm_sale_order_detail.order_id`

`demo_sm_product.id = demo_sm_sale_order_detail.product_id`

`demo_sm_product_type.id = demo_sm_product.type_id`

设置完的界面如下图：

| 表或视图                      | 关联类型 | 别名                     | 关联条件               |    |    |                    |
|---------------------------|------|------------------------|--------------------|----|----|--------------------|
| demo_sm_sale_order_detail | 主表   | demo_sm_sale_order_det |                    |    |    |                    |
| demo_sm_sale_order        | 左连接  | demo_sm_sale_order     | demo_sm_sale_order | id | 等于 | demo_sm_sale_order |
| demo_sm_product           | 左连接  | demo_sm_product        | demo_sm_product    | id | 等于 | demo_sm_sale_order |
| demo_sm_product_type      | 左连接  | demo_sm_product_type   | demo_sm_product_ty | id | 等于 | demo_sm_product    |

然后在下面的查询字段选项页里，点击添加字段按钮，添加 demo\_sm\_sale\_order\_detail 表的所有字段、demo\_sm\_sale\_order 表的状态和 order\_date 字段、demo\_sm\_product 表的 code、name 和 sort\_num 字段、demo\_sm\_product\_type 表的名字和 sort\_num 字段，并将 demo\_sm\_product 表的 code 字段的别名设置为 product\_code，name 字段的别名设置为 product\_name，sort\_num 字段的别名设置为 product\_sort\_num，将 demo\_sm\_product\_type 表的名字字段的别名设置为 product\_type\_name，sort\_num 字段的别名设置为 product\_type\_sort\_num，设置完的界面如下图所示：

| 查询字段 查询条件 分组条件 过滤条件                                                                                                                                |                       |                           |            |      |      |
|----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|---------------------------|------------|------|------|
| <input type="button" value="+ 添加字段"/> <input type="button" value="+ 添加自定义字段"/> <input type="button" value="- 删除字段"/> <input type="checkbox"/> 排除重复 |                       |                           |            |      |      |
| <input type="checkbox"/>                                                                                                                           | 字段别名                  |                           | 字段格式       |      | 字段备注 |
| <input type="checkbox"/>                                                                                                                           | quantity              | demo_sm_sale_order_detail | quantity   | 数量   |      |
| <input type="checkbox"/>                                                                                                                           | unit                  | demo_sm_sale_order_detail | unit       | 计量单位 |      |
| <input type="checkbox"/>                                                                                                                           | amount                | demo_sm_sale_order_detail | amount     | 金额   |      |
| <input type="checkbox"/>                                                                                                                           | comment               | demo_sm_sale_order_detail | comment    | 备注   |      |
| <input type="checkbox"/>                                                                                                                           | status                | demo_sm_sale_order        | status     | 状态   |      |
| <input type="checkbox"/>                                                                                                                           | order_date            | demo_sm_sale_order        | order_date | 下单日期 |      |
| <input type="checkbox"/>                                                                                                                           | product_code          | demo_sm_product           | code       | 产品编码 |      |
| <input type="checkbox"/>                                                                                                                           | product_name          | demo_sm_product           | name       | 产品名称 |      |
| <input type="checkbox"/>                                                                                                                           | product_type_name     | demo_sm_product_type      | name       | 分类名称 |      |
| <input type="checkbox"/>                                                                                                                           | product_sort_num      | demo_sm_product           | sort_num   | 产品序号 |      |
| <input type="checkbox"/>                                                                                                                           | product_type_sort_num | demo_sm_product_type      | sort_num   | 分类序号 |      |

都设置好后，点击工具栏的保存按钮，在弹出的提示对话框中点确定，保存该视图的数据模型，并同步到数据库。

## 6.3 实现按客户统计销售情况

下面，我们来实现客户订单情况统计功能，希望通过该功能可以汇总统计各个客户的订单总数量、订单金额合计、优惠金额合计和实收金额合计，并且支持按起止时间统计。

### 6.3.1 创建按客户统计销售情况的后台服务

在销售管理示例项目的后台服务中，点击 customer 文件夹右边的加号，打开如下图所示的“后台资源创建向导”对话框：



在此，选中“数据统计服务”，将资源名称设置为：statBySaleOrder，然后点下一步，在

打开的对话框中，先选择前面创建的销售订单主表视图的数据模型 `vdemo_sm_sale_order.tdm`，然后将服务说明设置为：统计客户销售订单情况，然后在下面的“统计字段”选项页里，按照下表所示添加四个字段，并设置对应的统计函数：

| 字段别名            | 统计函数 | 统计字段或条件         | 备注                                    |
|-----------------|------|-----------------|---------------------------------------|
| customer_name   | 不计算  | customer_name   | 不进行任何统计运算，直接显示客户名称                    |
| order_count     | 求总数  | id              | 计算各个客户的订单总数量，按 id 字段统计，因为 id 是主键，不会重复 |
| order_amount    | 求合计值 | order_amount    | 计算各个客户的订单金额合计值                        |
| discount_amount |      | discount_amount | 计算各个客户的优惠金额合计值                        |
| actual_amount   | 求合计值 | actual_amount   | 计算各个客户的实收金额合计值                        |

设置完的界面如下图所示：



然后点“动态查询条件”选项卡，在此在查询字段列表中选中下单日期 `order_date` 字段，查询条件选择起止日期，然后点加号，即可添加开始日期 `start_date` 和结束日期 `end_date` 两个动态查询条件，当传递这两个参数时，就会执行指定的查询表达式，设置完的界面如下图所示：



创建数据统计服务

服务路径: /app/demo\_sm/service/customer/statBySaleOrder.tbs

返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order.tdm

选择...

服务说明: 统计客户销售订单情况

编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段

动态查询条件

固定查询条件

分组条件

过滤条件

排序规则

查询字段:

查询条件:

+

| 请求参数名称     | 参数说明 | 查询表达式(SQL, 双击可编辑)            |   |
|------------|------|------------------------------|---|
| start_date | 开始日期 | order_date>={req.start_date} | - |
| end_date   | 结束日期 | order_date<={req.end_date}   | - |
|            |      |                              |   |
|            |      |                              |   |
|            |      |                              |   |

确定

取消

然后点击“固定查询条件”选项卡，将查询条件设置为：status=1，表示仅查询审核通过的订单，设置完的界面如下图所示：

创建数据统计服务

服务路径: /app/demo\_sm/service/customer/statBySaleOrder.tbs

返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order.tdm

选择...

服务说明: 统计客户销售订单情况

编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段

动态查询条件

固定查询条件

分组条件

过滤条件

排序规则

查询字段:

查询条件:

+

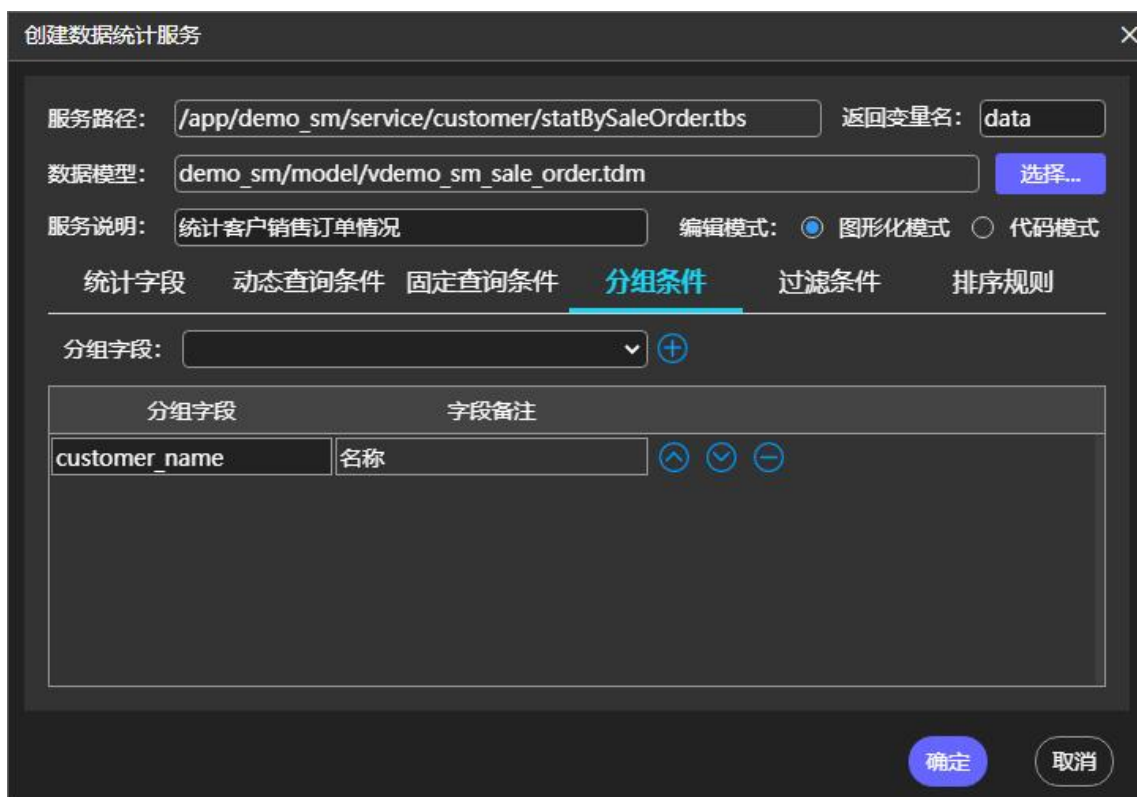
status=1

确定

取消



然后点击“分组条件”选项卡，将 customer\_name 字段添加为分组条件，表示按照客户名称进行分组统计，设置界面如下图所示：



都设置好后，点确定按钮，即可完成该后台服务的创建，TaskBuilder 会自动打开该服务，然后继续进行完善，需要在该服务中添加起止日期变量的声明和设置语句，界面如下图：



### 6.3.2 创建按客户统计销售情况的前端页面

在销售管理示例项目的前端文件中，点击 customer 文件夹右边的加号，打开如下图所示的“前端资源创建向导”对话框：



在该对话框中选中“数据查询 TFP”，资源名称设置为 statBySaleOrder，然后点下一步：



在弹出的“创建数据查询 TFP”对话框中，先选择销售订单主表视图数据模型：

vdemo\_sm\_sale\_order.tdm，页面标题设置为：客户订单情况统计，不启用翻页，不显示添加、

修改和删除按钮，数据查询服务使用已有的服务，就是上面创建的客户订单统计后台服务：

demo\_sm/service/customer/statBySaleOrder.tbs，然后点下一步显示如下界面：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段

添加关联字段

|                                     | 字段名称            | 字段备注  | 字段格式(双击编辑)      | 关联数据字典(双击选择) | 字段别名        |
|-------------------------------------|-----------------|-------|-----------------|--------------|-------------|
| <input checked="" type="checkbox"/> | id              | 订单编号  | id              |              | order_count |
| <input type="checkbox"/>            | code            | 订单编码  | code            |              |             |
| <input type="checkbox"/>            | operator_id     | 经办人编号 | operator_id     |              |             |
| <input type="checkbox"/>            | order_date      | 下单日期  | order_date      |              |             |
| <input type="checkbox"/>            | customer_id     | 客户编号  | customer_id     |              |             |
| <input type="checkbox"/>            | contacts        | 联系人   | contacts        |              |             |
| <input type="checkbox"/>            | contact_number  | 联系电话  | contact_number  |              |             |
| <input checked="" type="checkbox"/> | order_amount    | 订单金额  | order_amount    |              |             |
| <input checked="" type="checkbox"/> | discount_amount | 优惠金额  | discount_amount |              |             |
| <input checked="" type="checkbox"/> | actual_amount   | 实收金额  | actual_amount   |              |             |
| <input type="checkbox"/>            | creator_id      | 创建人编号 | creator_id      |              |             |
| <input type="checkbox"/>            | create_time     | 创建时间  | create_time     |              |             |

上一步

下一步

取消

在该界面里选中 id、order\_amount、discount\_amount、actual\_amount、customer\_name 这几个字段，并将 id 字段的别名设置为：order\_count，与上面的统计服务里设置的要一致，然后点下一步，打开如下所示界面：

创建数据查询TFP

第三步：设置显示格式

+

| 列名   | 列类型 | 数据绑定格式(js表达式)     | 文本显示格式 | 对齐方式 | 宽度  | 文字颜色 | 加粗 | 求和 |
|------|-----|-------------------|--------|------|-----|------|----|----|
| 客户名称 | 文本  | {customer_name}   |        | 左对齐  | 300 |      |    |    |
| 订单数量 | 文本  | {order_count}     |        | 居中对齐 | 100 |      |    |    |
| 订单金额 | 文本  | {order_amount}    | 金额     | 右对齐  | 120 |      |    |    |
| 优惠金额 | 文本  | {discount_amount} | 金额     | 右对齐  | 120 |      |    |    |
| 实收金额 | 文本  | {actual_amount}   | 金额     | 右对齐  | 120 |      |    |    |

上一步

下一步

取消

在此，参考上图移动字段的显示顺序，并设置好各个字段的列名、文本显示格式、对其方式、宽度和合计等参数，然后点下一步，打开如下所示界面：

创建数据查询TFP

第四步：设置动态查询参数

查询字段：  查询条件：

| 参数名称       | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)                  | 默认值 | 数据字典(双击选择) | 可选值                              |
|------------|------|-------|----|------------------------------|-----|------------|----------------------------------|
| start_date | 开始日期 | 日期    |    | order_date>={req.start_date} |     |            | <input type="button" value="⊖"/> |
| end_date   | 结束日期 | 日期    |    | order_date<={req.end_date}   |     |            | <input type="button" value="⊖"/> |

在此界面中，在查询字段列表中选中下单日期字段（order\_date），查询条件选择起止日期，然后点添加按钮，添加两个根据起止日期查询的动态查询条件，然后点下一步，后面两步不需要设置任何参数，到最后一步点确定即可完成该页面的创建，然后点预览即可查看效果。

刷新 本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

开始日期：  结束日期：

| 产品分类 | 产品名称 | 数量 | 金额        |
|------|------|----|-----------|
| 日用品  | 拖把   | 20 | 14,638.00 |
| 日用品  | 电风扇  | 60 | 61,740.00 |
| 食品   | 方便面  | 4  | 60.00     |
| 合计   |      | 84 | 76,438.00 |

## 6.4 实现按产品统计销售情况

下面我们实现按产品统计销售情况的功能，可以统计各个产品的销售数量、销售总金额，并且能按产品分类进行分组和小计。

### 6.4.1 创建按产品统计销售情况的后台服务

在销售管理示例项目的后台服务中，点击 product 文件夹右边的加号，打开如下图所示的“后台资源创建向导”对话框，选中“数据统计服务”，然后将资源名称设置为：statByType。



然后点下一步，打开如下图所示的“创建数据统计服务”对话框，在此，先选择销售订单明细视图：vdemo\_sm\_sale\_order\_detail.tdm，然后将服务说明设置为：统计产品分类销售情况，然后参考上图，分别添加 product\_type\_name、product\_name、quantity 和 amount 这四个字段作为统计字段，其中前两个字段不进行任何计算，直接查询显示，后面两个字段设置为计算合计值。



创建数据统计服务

服务路径: /app/demo\_sm/service/product/statByType.tbs

返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm

选择...

服务说明: 统计产品分类销售情况

编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段

动态查询条件

固定查询条件

分组条件

过滤条件

排序规则

统计字段:

统计函数: 不计算

+

| 字段别名              | 统计函数 | 统计字段或条件(SQL, 双击可编辑) |   |
|-------------------|------|---------------------|---|
| product_type_name | 不计算  | product_type_name   | - |
| product_name      | 不计算  | product_name        | - |
| quantity          | 求合计值 | quantity            | - |
| amount            | 求合计值 | amount              | - |

确定

取消

然后点“动态查询条件”选项卡，在此在查询字段列表中选中下单日期 order\_date 字段，查询条件选择起止日期，然后点加号，即可添加开始日期 start\_date 和结束日期 end\_date 两个动态查询条件，当传递这两个参数时，就会执行指定的查询表达式，设置完的界面如下图所示：

创建数据统计服务

服务路径: /app/demo\_sm/service/product/statByType.tbs

返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm

选择...

服务说明: 统计产品分类销售情况

编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段

动态查询条件

固定查询条件

分组条件

过滤条件

排序规则

查询字段:

查询条件:

+

| 请求参数名称     | 参数说明 | 查询表达式(SQL, 双击可编辑)             |   |
|------------|------|-------------------------------|---|
| start_date | 开始日期 | order_date >={req.start_date} | - |
| end_date   | 结束日期 | order_date <={req.end_date}   | - |

确定

取消



然后点“固定查询条件”选项卡，在此设置查询条件：status=1，设置界面如下图：

The screenshot shows the 'Create Data Statistics Service' dialog box with the 'Fixed Query Conditions' tab selected. The 'Service Path' is '/app/demo\_sm/service/product/statByType.tbs' and the 'Return Variable Name' is 'data'. The 'Data Model' is 'demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm'. The 'Service Description' is '统计产品分类销售情况'. The 'Edit Mode' is 'Graphical Mode'. The 'Query Fields' tab is selected, and the 'Query Conditions' dropdown is set to 'status=1'. The 'Confirm' button is highlighted.

| 统计字段     | 动态查询条件 | 固定查询条件 | 分组条件 | 过滤条件 | 排序规则 |
|----------|--------|--------|------|------|------|
| 查询字段:    |        | 查询条件:  |      |      |      |
| status=1 |        |        |      |      |      |

然后点击“分组条件”选项卡，在此分别将 product\_type\_name 和 product\_name 字段添加为分组字段，界面如下图所示：

The screenshot shows the 'Create Data Statistics Service' dialog box with the 'Grouping Conditions' tab selected. The 'Service Path' is '/app/demo\_sm/service/product/statByType.tbs' and the 'Return Variable Name' is 'data'. The 'Data Model' is 'demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm'. The 'Service Description' is '统计产品分类销售情况'. The 'Edit Mode' is 'Graphical Mode'. The 'Grouping Fields' dropdown is set to '+'. The 'Grouping Conditions' table is shown below.

| 分组字段              | 字段备注 |          |
|-------------------|------|----------|
| product_type_name | 分类名称 | ⬆️ ⬆️ ⬆️ |
| product_name      | 产品名称 | ⬆️ ⬆️ ⬆️ |

然后点击“排序规则”选项卡，在此分别将 product\_type\_sort\_num 和 product\_sort\_num 字段添加为排序字段，排序规则都选正序，设置界面如下图所示：

创建数据统计服务

服务路径: /app/demo\_sm/service/product/statByType.tbs 返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm 选择...

服务说明: 统计产品分类销售情况 编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段 动态查询条件 固定查询条件 分组条件 过滤条件 **排序规则**

排序字段: 排序规则: 正序

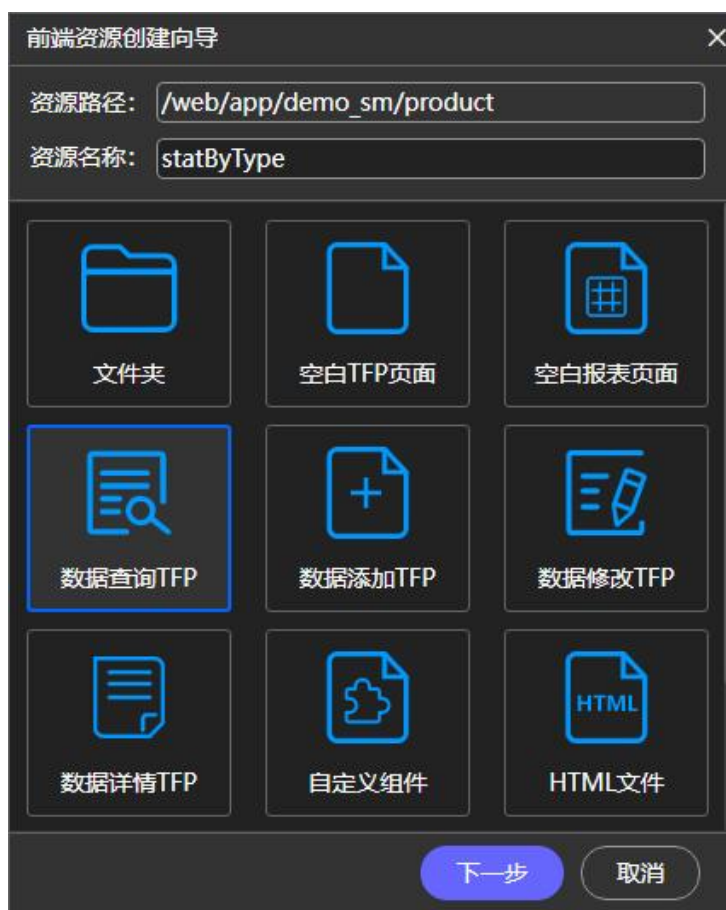
| 排序字段                  | 字段备注 | 排序规则 |       |
|-----------------------|------|------|-------|
| product_type_sort_num | 分类序号 | asc  | ⬆ ⬇ ⬇ |
| product_sort_num      | 产品序号 | asc  | ⬆ ⬇ ⬇ |

确定 取消

最后点击确定按钮，即可完成该统计服务的创建。

## 6.4.2 创建按产品统计销售情况的前端页面

在销售管理示例项目的前端文件中，点击 product 文件夹右边的加号，打开“前端资源创建向导”对话框，在该对话框中选中“数据查询 TFP”，资源名称设置为 statByType，界面如下图所示：



然后点下一步，弹出下图所示的“创建数据查询 TFP”对话框：



在此，先选择销售订单明细视图数据模型：vdemo\_sm\_sale\_order\_detail.tdm，页面标题设

置为：产品分类销售情况统计，不启用翻页，不显示添加、修改和删除按钮，数据查询服务使用已有的服务，就是上面创建的后台服务：demo\_sm/service/product/statByType.tbs，然后点下一步，显示如下界面：

创建数据查询TFP

第二步：设置要查询的字段

添加自定义字段

添加关联字段

|                                     | 字段名称                  | 字段备注 | 字段格式(双击编辑)            | 关联数据字典(双击选择) | 字段别名 |
|-------------------------------------|-----------------------|------|-----------------------|--------------|------|
| <input type="checkbox"/>            | price                 | 单价   | price                 |              |      |
| <input checked="" type="checkbox"/> | quantity              | 数量   | quantity              |              |      |
| <input type="checkbox"/>            | unit                  | 计量单位 | unit                  |              |      |
| <input checked="" type="checkbox"/> | amount                | 金额   | amount                |              |      |
| <input type="checkbox"/>            | comment               | 备注   | comment               |              |      |
| <input type="checkbox"/>            | status                | 状态   | status                |              |      |
| <input type="checkbox"/>            | order_date            | 下单日期 | order_date            |              |      |
| <input type="checkbox"/>            | product_code          | 产品编码 | product_code          |              |      |
| <input checked="" type="checkbox"/> | product_name          | 产品名称 | product_name          |              |      |
| <input checked="" type="checkbox"/> | product_type_name     | 分类名称 | product_type_name     |              |      |
| <input type="checkbox"/>            | product_sort_num      | 产品序号 | product_sort_num      |              |      |
| <input type="checkbox"/>            | product_type_sort_num | 分类序号 | product_type_sort_num |              |      |

上一步

下一步

取消

在此，选中 quantity、amount、product\_name 和 product\_type\_name 这四个字段，然后点击下一步，显示如下界面：

创建数据查询TFP

第三步：设置显示格式

+

| 列名   | 列类型 | 数据绑定格式(js表达式)       | 文本显示格式 | 对齐方式 | 宽度  | 文字颜色 | 加粗 | 求和 |
|------|-----|---------------------|--------|------|-----|------|----|----|
| 产品分类 | 文本  | {product_type_name} |        | 居中对齐 | 100 |      |    |    |
| 产品名称 | 文本  | {product_name}      |        | 居中对齐 | 200 |      |    |    |
| 数量   | 文本  | {quantity}          |        | 居中对齐 | 100 |      |    |    |
| 金额   | 文本  | {amount}            | 金额     | 右对齐  | 100 |      |    |    |

上一步

下一步

取消

在此，参考上图移动字段的显示顺序，并设置好各个字段的列名、文本显示格式、对其方式、宽度和求和等参数，然后点下一步，打开如下所示界面：

创建数据查询TFP

第四步：设置动态查询参数

查询字段：

查询条件：

+

| 参数名称       | 参数说明 | 输入项类型 | 宽度 | 查询表达式(双击编辑)                  | 默认值 | 数据字典(双击选择) | 可选值          |
|------------|------|-------|----|------------------------------|-----|------------|--------------|
| start_date | 开始日期 | 日期    |    | order_date>={req.start_date} |     |            | <div>⊖</div> |
| end_date   | 结束日期 | 日期    |    | order_date<={req.end_date}   |     |            | <div>⊖</div> |

上一步

下一步

取消

在此界面中，在查询字段列表选中下单日期字段（order\_date），查询条件选择起止日期，然后点添加按钮，添加两个根据起止日期查询的动态查询条件，然后点下一步，后面两步不需要设置任何参数，到最后一步点确定即可完成该页面的创建，然后点预览即可查看效果。

刷新

本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

开始日期：结束日期：

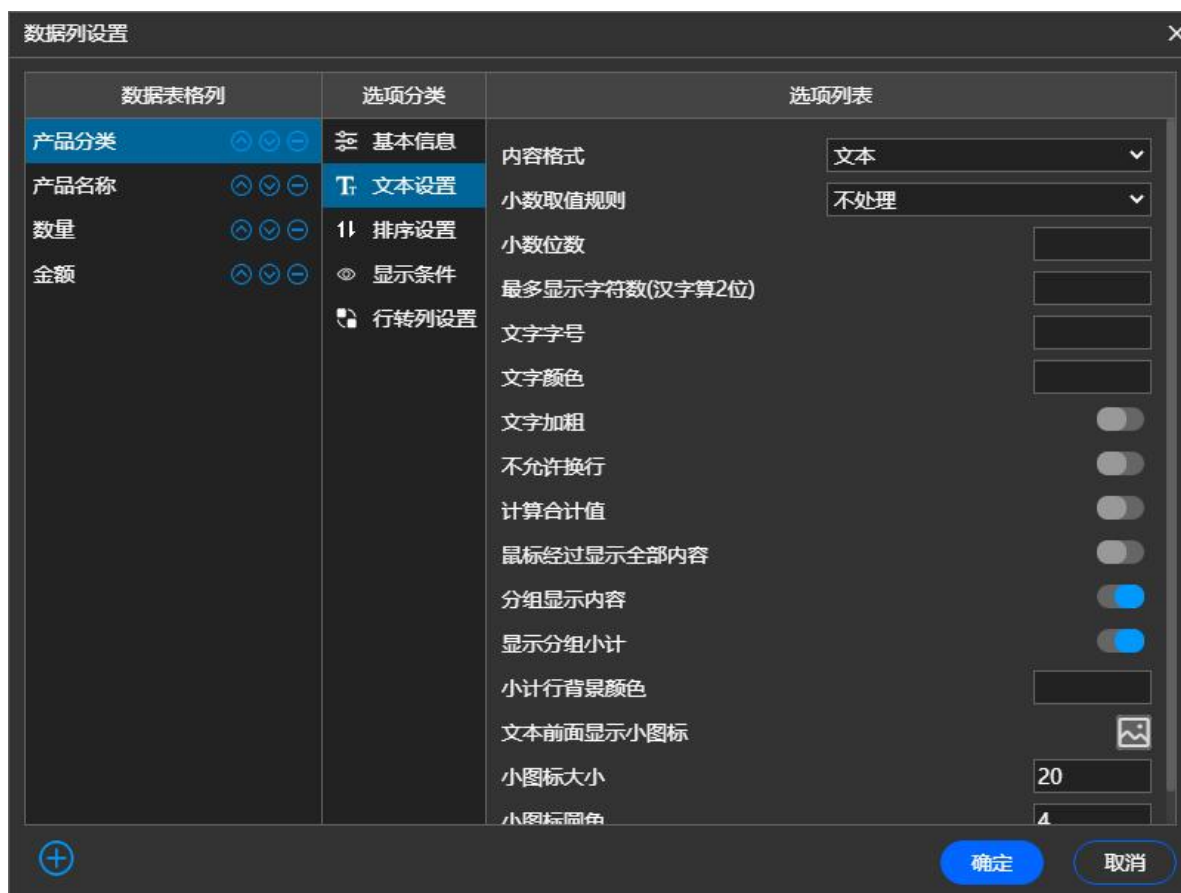
Q 查询

| 产品分类 | 产品名称 | 数量 | 金额        |
|------|------|----|-----------|
| 日用品  | 拖把   | 20 | 14,638.00 |
| 日用品  | 电风扇  | 60 | 61,740.00 |
| 食品   | 方便面  | 4  | 60.00     |
| 合计   |      | 84 | 76,438.00 |

### 6.4.3 实现按产品类别分组统计

我们再完善一下上面的统计页面，实现按产品分类分组显示并进行分组小计的功能，操作方法如下：

在上面的页面中选中数据表格组件，点击属性设置里的“数据列设置”，在弹出的对话框中，点击产品分类字段的文本设置，把分组显示内容和显示分组小计两个开关打开，如下图：



然后点击确定，然后保存页面，点击预览，即可查看效果，如下所示：



刷新

本窗口仅供开发时预览参考，实际效果请在任掌后台管理系统内查看。

开始日期：结束日期：

查询

| 产品分类 | 产品名称        | 数量  | 金额         |
|------|-------------|-----|------------|
| 电器   | 电热水壶        | 20  | 2,400.00   |
|      | 电风扇         | 80  | 65,520.00  |
|      | 小米54英寸电视    | 10  | 38,000.00  |
|      | 海尔5升滚筒洗衣机   | 10  | 35,000.00  |
|      | 格力2P壁挂空调    | 10  | 36,000.00  |
| 小计   |             | 130 | 176,920.00 |
| 日用品  | 拖把          | 30  | 15,357.00  |
|      | 300ML不锈钢保温杯 | 10  | 0.00       |
| 小计   |             | 40  | 15,357.00  |
| 食品   | 方便面         | 14  | 210.00     |
|      | 10kg装五常大米   | 10  | 900.00     |
| 小计   |             | 24  | 1,110.00   |
| 合计   |             | 194 | 193,387.00 |

## 6.5 实现按日期统计产品分类销售情况

接下来我们实现按日期统计各个产品分类销售情况的功能。

### 6.5.1 创建按日期统计产品分类销售情况的后台服务

在销售管理示例项目的后台服务中，点击 product 文件夹右边的加号，打开如下图所示的“后台资源创建向导”对话框，选中“数据统计服务”，然后将资源名称设置为：statByDate。



然后点下一步，打开如下图所示的“创建数据统计服务”对话框，在此，先选择销售订单明细视图：vdemo\_sm\_sale\_order\_detail.tdm，然后将服务说明设置为：按日期统计各产品分类销售情况，然后分别添加 order\_date、product\_type\_name、quantity 和 amount 这 4 个字段作为统计字段，其中前两个字段不进行任何计算，直接显示，后面两个字段设置为计算合计值。

创建数据统计服务

服务路径: /app/demo\_sm/service/product/statByDate.tbs

返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm

选择...

服务说明: 按日期统计各产品分类销售情况

编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段

动态查询条件

固定查询条件

分组条件

过滤条件

排序规则

统计字段:

统计函数: 不计算

+

| 字段别名              | 统计函数 | 统计字段或条件(SQL, 双击可编辑) |   |
|-------------------|------|---------------------|---|
| order_date        | 不计算  | order_date          | - |
| product_type_name | 不计算  | product_type_name   | - |
| quantity          | 求合计值 | quantity            | - |
| amount            | 求合计值 | amount              | - |

确定

取消

然后点“动态查询条件”选项卡，在此在查询字段列表中选中 product\_type\_name 字段，查询条件选择模糊查询，然后点加号，即可添加一个按产品分类名称模糊查询的动态查询条件，当前端传递这个参数时，就会按指定的查询表达式查询，设置完的界面如下图所示：

创建数据统计服务

服务路径: /app/demo\_sm/service/product/statByDate.tbs

返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm

选择...

服务说明: 按日期统计各产品分类销售情况

编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段

动态查询条件

固定查询条件

分组条件

过滤条件

排序规则

查询字段:

查询条件:

+

| 请求参数名称            | 参数说明               | 查询表达式(SQL, 双击可编辑)                             |   |
|-------------------|--------------------|-----------------------------------------------|---|
| product_type_name | product_type_name, | product_type_name like {%req.product_type_nar | - |

确定

取消

然后点“固定查询条件”选项卡，在此设置查询条件：status=1，设置界面如下图：

The screenshot shows the 'Create Data Statistics Service' dialog box with the 'Fixed Query Conditions' tab selected. The 'Query Conditions' field contains 'status=1'. The 'Query Fields' field is empty. The 'Service Path' is '/app/demo\_sm/service/product/statByDate.tbs' and the 'Return Variable Name' is 'data'. The 'Data Model' is 'demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm'. The 'Service Description' is '按日期统计各产品分类销售情况'. The 'Edit Mode' is 'Graphical Mode'. The 'Query Fields' and 'Query Conditions' fields are at the top. Below them is a large text area for the query conditions, containing 'status=1'. At the bottom are '确定' (Confirm) and '取消' (Cancel) buttons.

| 统计字段     | 动态查询条件 | 固定查询条件 | 分组条件 | 过滤条件 | 排序规则 |
|----------|--------|--------|------|------|------|
| 查询字段:    |        | 查询条件:  |      |      |      |
| status=1 |        |        |      |      |      |

然后点击“分组条件”选项卡，在此分别将 order\_date 和 product\_type\_name 字段添加为分组字段，界面如下图所示：

The screenshot shows the 'Create Data Statistics Service' dialog box with the 'Grouping Conditions' tab selected. The 'Grouping Fields' field is empty. The 'Service Path' is '/app/demo\_sm/service/product/statByDate.tbs' and the 'Return Variable Name' is 'data'. The 'Data Model' is 'demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm'. The 'Service Description' is '按日期统计各产品分类销售情况'. The 'Edit Mode' is 'Graphical Mode'. The 'Grouping Fields' field is at the top. Below it is a table with two columns: 'Grouping Fields' and 'Field Remarks'. The table contains two rows: 'order\_date' with '下单日期' and 'product\_type\_name' with '分类名称'. At the bottom are '确定' (Confirm) and '取消' (Cancel) buttons.

| 统计字段              | 动态查询条件 | 固定查询条件 | 分组条件 | 过滤条件 | 排序规则 |
|-------------------|--------|--------|------|------|------|
| 分组字段:             |        |        |      |      |      |
| order_date        |        |        | 下单日期 |      |      |
| product_type_name |        |        | 分类名称 |      |      |

然后点击“排序规则”选项卡，在此分别将 order\_date 和 product\_type\_sort\_num 字段添加为排序字段，排序规则都选正序，设置界面如下图所示：

创建数据统计服务

服务路径: /app/demo\_sm/service/product/statByDate.tbs 返回变量名: data

数据模型: demo\_sm/model/vdemo\_sm\_sale\_order\_detail.tdm 选择...

服务说明: 按日期统计各产品分类销售情况 编辑模式: ☒ 图形化模式 ☐ 代码模式

统计字段 动态查询条件 固定查询条件 分组条件 过滤条件 **排序规则**

排序字段: 排序规则: 正序

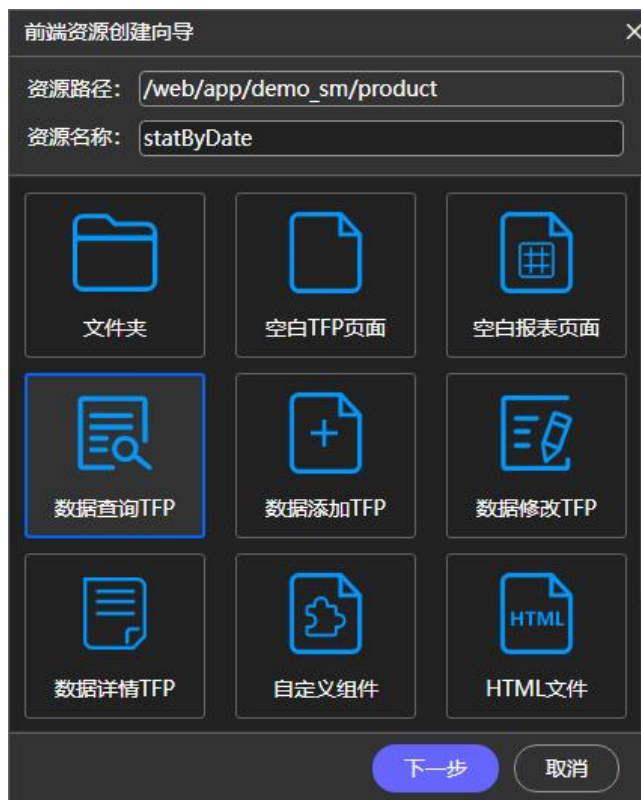
| 排序字段                  | 字段备注 | 排序规则 |       |
|-----------------------|------|------|-------|
| order_date            | 下单日期 | asc  | ⬆ ⬇ ⬇ |
| product_type_sort_num | 分类序号 | asc  | ⬆ ⬇ ⬇ |

确定 取消

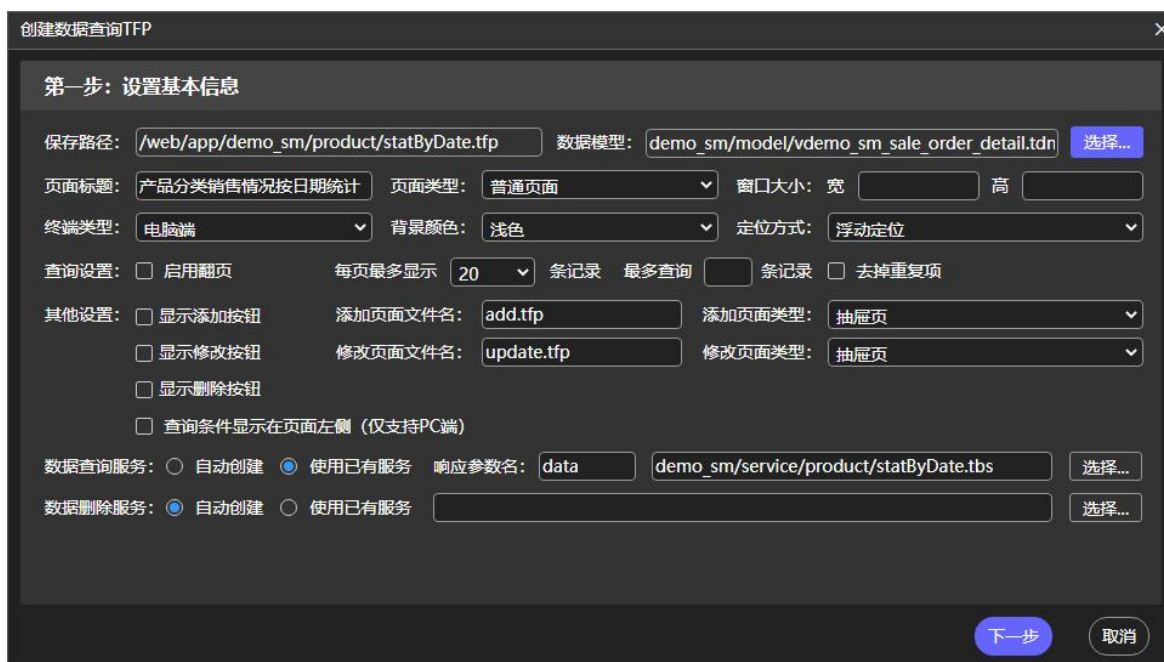
最后点击确定按钮，即可完成该统计服务的创建。

## 6.5.2 创建按日期统计产品分类销售情况的前端页面

在销售管理示例项目的前端文件中，点击 product 文件夹右边的加号，打开如下图所示的“前端资源创建向导”对话框，在该对话框中选中“数据查询 TFP”，资源名称设置为 statByDate：



然后点下一步，弹出下图所示的“创建数据查询 TFP”对话框：

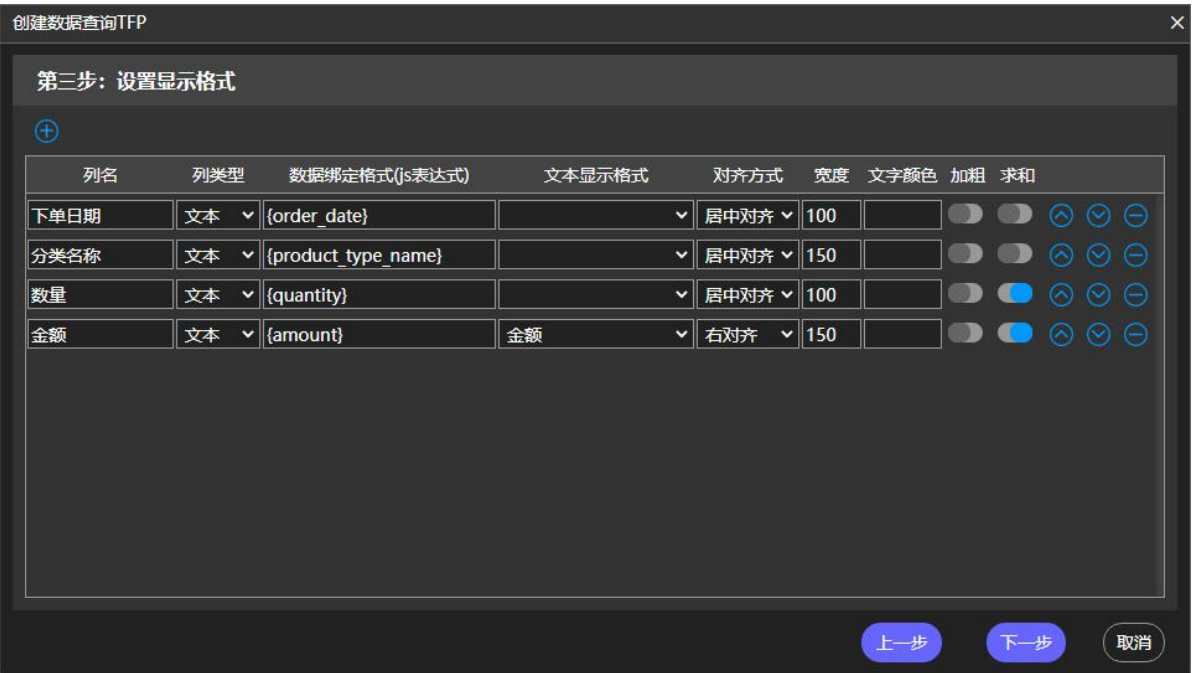


在此，先选择销售订单明细视图数据模型：vdemo\_sm\_sale\_order\_detail.tdm，页面标题设置为：产品分类销售情况按日期统计，不启用翻页，不显示添加、修改和删除按钮，数据查询服务使用上面创建的后台服务：demo\_sm/service/product/statByDate.tbs，然后点下一步。





在此，选中 quantity、amount、order\_date 和 product\_type\_name 这四个字段，然后点下一步，显示如下界面：

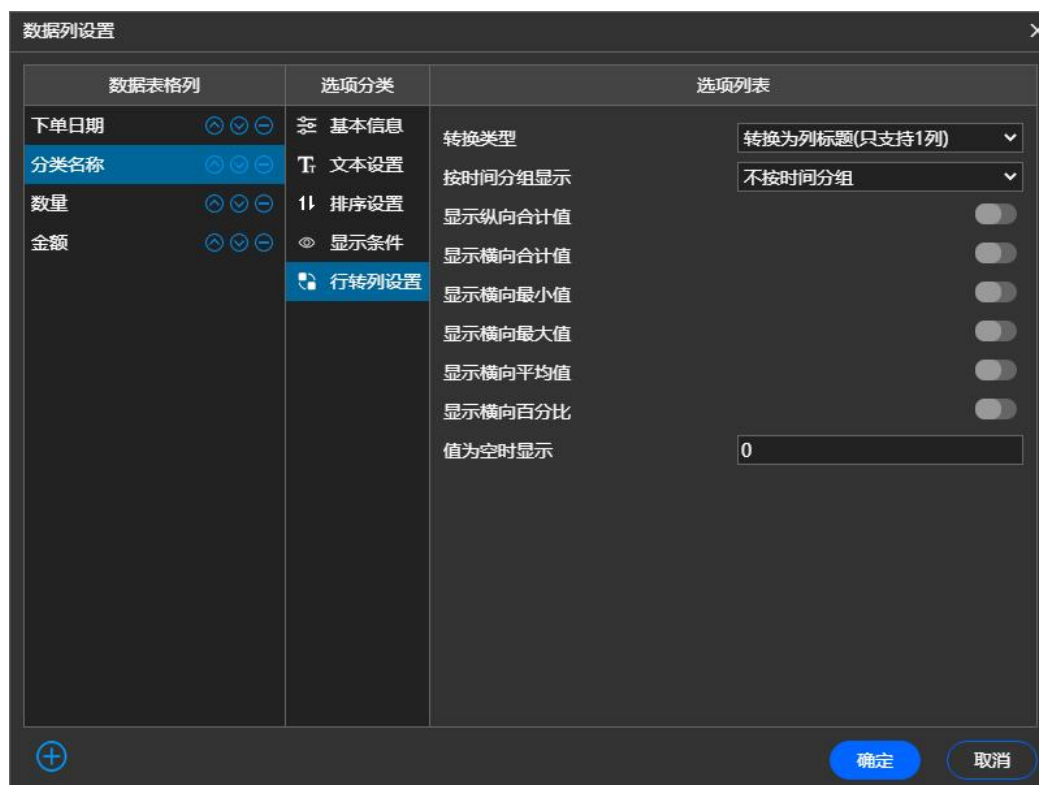


在此，参考上图移动字段的显示顺序，并设置好各个字段的列名、文本显示格式、对其方式、宽度和求和等参数，然后点下一步，打开如下所示界面：



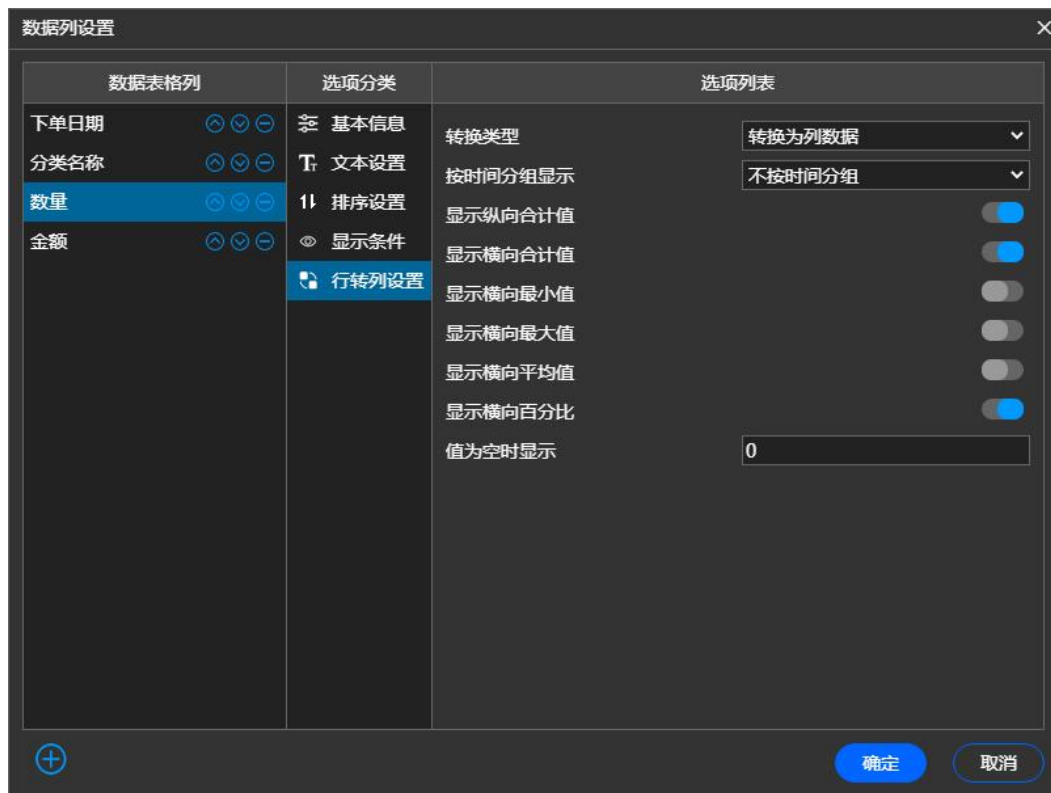
在此界面中，在查询字段列表中选中下单日期字段（order\_date），查询条件选择起止日期，然后点添加按钮，添加两个根据起止日期查询的动态查询条件，然后点下一步，后面两步不需要设置任何参数，到最后一步点确定即可完成该页面的创建。

然后选中该页面中的数据表格组件，点击属性设置里的“数据列设置”，打开如下对话框：

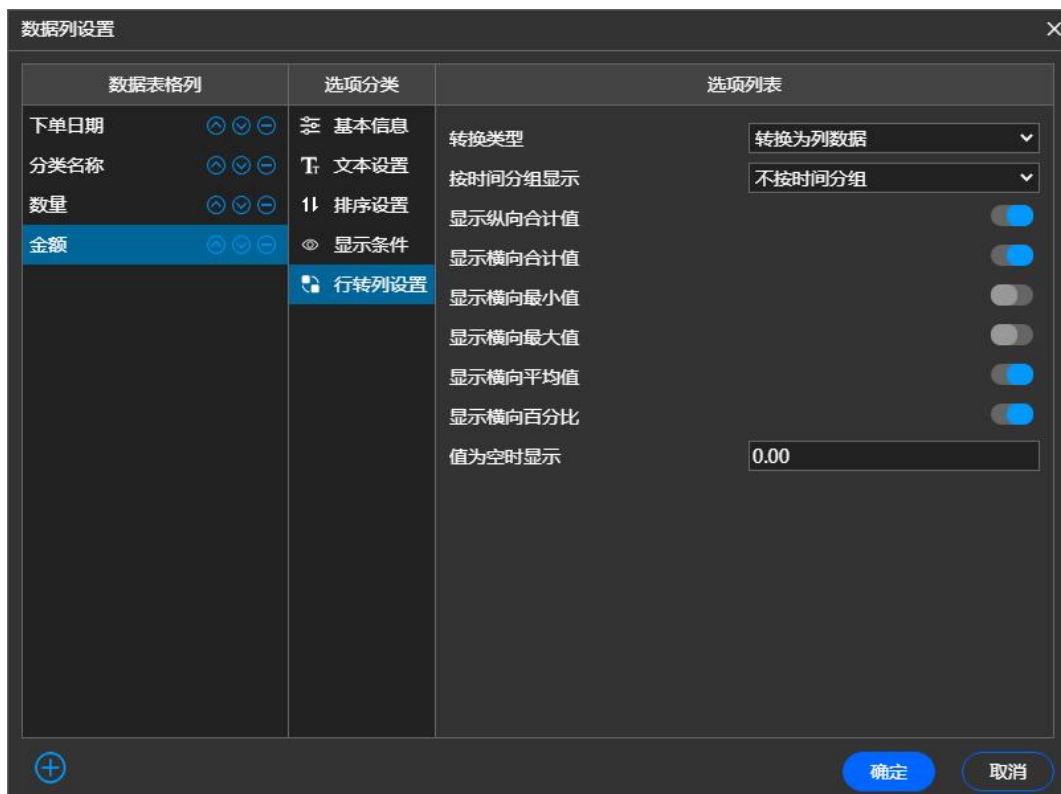


在该对话框中，先在左侧列表中选中“分类名称”列，然后点击中间的“行转列设置”，

将转换类型设置为“转换为列标题”，然后再选中“数量”列，还是点击中间的“行转列设置”，将显示纵向合计值、显示横向合计值和显示横向百分比开关打开，如下图所示：



然后再选中“金额”列，同样是点击中间的“行转列设置”：



然后将显示纵向合计值、显示横向合计值、显示横向平均值和显示横向百分比开关打开。

都设置好后点确定按钮，然后保存页面，然后点击预览，可以查看运行效果，如下图所示：

刷新

本窗口仅供开发时预览参考，实际效果请在任擎后台管理系统内查看。

| 下单日期       | 电器  |            | 食品 |          | 日用品 |           | 数量合计 | 数量占比  | 金额合计       | 金额平均      | 金额占比  |
|------------|-----|------------|----|----------|-----|-----------|------|-------|------------|-----------|-------|
|            | 数量  | 金额         | 数量 | 金额       | 数量  | 金额        |      |       |            |           |       |
| 2024-08-18 | 20  | 3,780.00   | 4  | 60.00    | 0   | 0.00      | 24   | 12.4% | 3,840.00   | 1,280.00  | 2.0%  |
| 2024-08-19 | 20  | 3,090.00   | 0  | 0.00     | 0   | 0.00      | 20   | 10.3% | 3,090.00   | 1,030.00  | 1.6%  |
| 2024-08-21 | 50  | 112,090.00 | 20 | 1,050.00 | 20  | 719.00    | 90   | 46.4% | 113,859.00 | 37,953.00 | 58.9% |
| 2024-08-24 | 40  | 57,960.00  | 0  | 0.00     | 20  | 14,638.00 | 60   | 30.9% | 72,598.00  | 24,199.33 | 37.5% |
|            |     |            |    |          |     |           |      |       |            |           |       |
| 合计         | 130 | 176,920.00 | 24 | 1,110.00 | 40  | 15,357.00 | 194  |       | 193,387.00 |           |       |

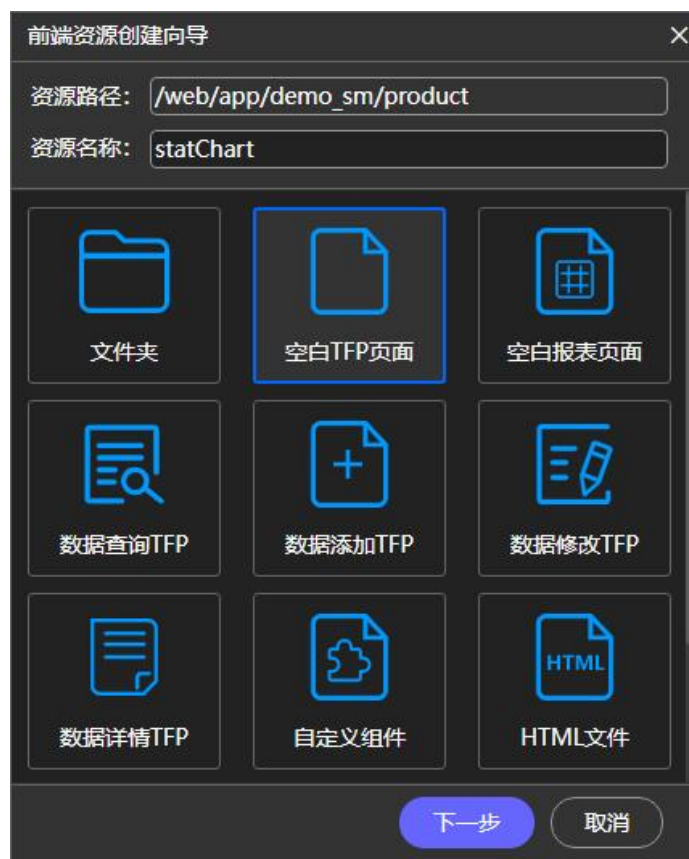
由上图可以看到，本来作为行数据的产品分类变成了统计表内的列标题，这种情况适用于某一系列数据是固定的几大类，然后想在统计报表里横向显示的情况。

## 6.6 用统计图显示产品销售情况

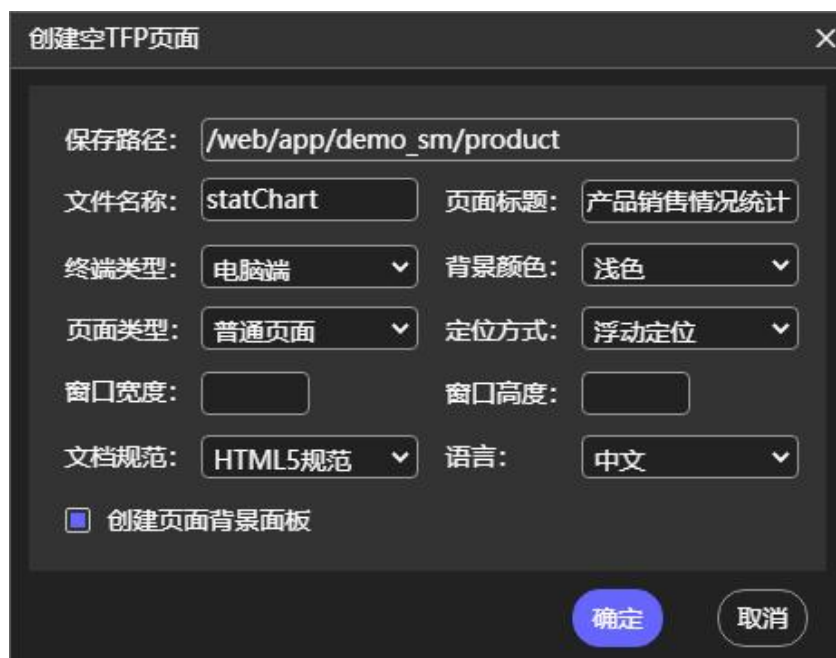
下面，教大家怎么用 TaskBuilder 提供的各种统计图表组件来实现产品销售情况的图形化展示，由于前面已经创建了统计产品销售情况的后台服务，这里就不用再建了，直接使用即可。

### 6.6.1 创建空白前端页面

在销售管理示例项目的前端文件中，点击 product 文件夹右边的加号，打开如下图所示的“前端资源创建向导”对话框，在该对话框中选中“空白 TFP 页面”，资源名称设置为 statChart：



然后点击下一步，打开如下对话框：



在此，将页面标题设置为：产品销售情况统计，选中“创建页面背景面板”，然后点下一步，即可创建一个空白的 tfp 页面。

然后从组件库的容器组件中，拖拽一个弹性栅格组件到页面上，会弹出以下对话框：

元素排列方向：  
☒ 横向 ☐ 纵向

元素数量：  
☒ 2 ☐ 3 ☐ 4

元素默认高度：  
400 px

奇偶数列宽度比例：  
☒ 1:1 ☐ 1:2 ☐ 1:3

边距：  
上: 5 下: 5 左: 10 右: 10

在这里，将元素默认高度设置为 400px，然后点确定，然后分别选中这个弹性栅格的左右两个面板，在属性里，将主轴对其方式都设置为居中对齐。

## 6.6.2 拖拽和设置柱状图

从组件库的统计图表中拖拽一个柱状图组件到弹性栅格的左侧面板中，然后点击属性设置里的数据查询设置，打开如下所示对话框：

数据查询设置

查询数据方式：  
☐ 数据模型 ☐ 数据表单 ☒ 后台服务 ☐ SQL语句 ☐ HTTP请求

**后台服务** 请求参数

☒ 直接从后台选择指定的后台服务 demo\_sm/service/product/statByType.tbs

☐ 使用页面上已有的服务请求组件

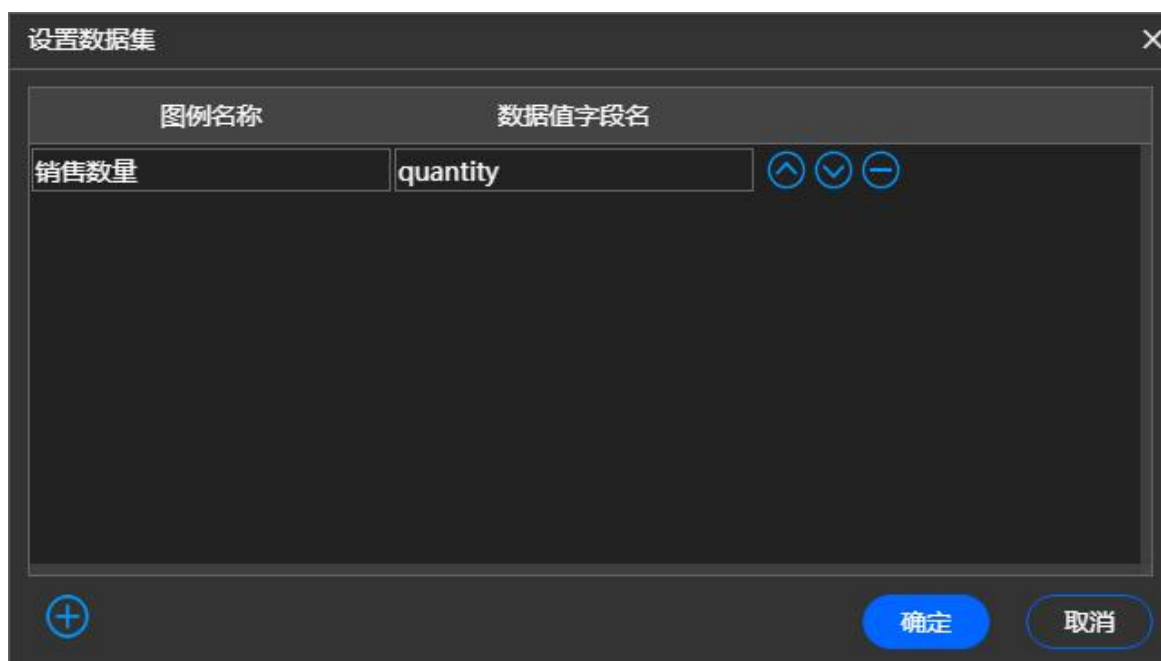
查询结果参数名: data



在该对话框中，首先将查询数据方式设置为“后台服务”，然后在下面点选择按钮，选择前面创建的统计产品销售情况的后台服务：demo\_sm/service/product/statByType.tbs，然后点确定，即可完成数据查询设置。

然后在这个柱状图组件的属性里，将分类数据字段设置为：product\_name，表示按产品名称进行分组显示。

然后再点击属性设置里的“数据值字段设置”，打开如下所示对话框：



在此，点击左下角的加号按钮添加一行，然后将图例名称设置为销售数量，数据值字段名设置为 quantity，然后点确定即可。

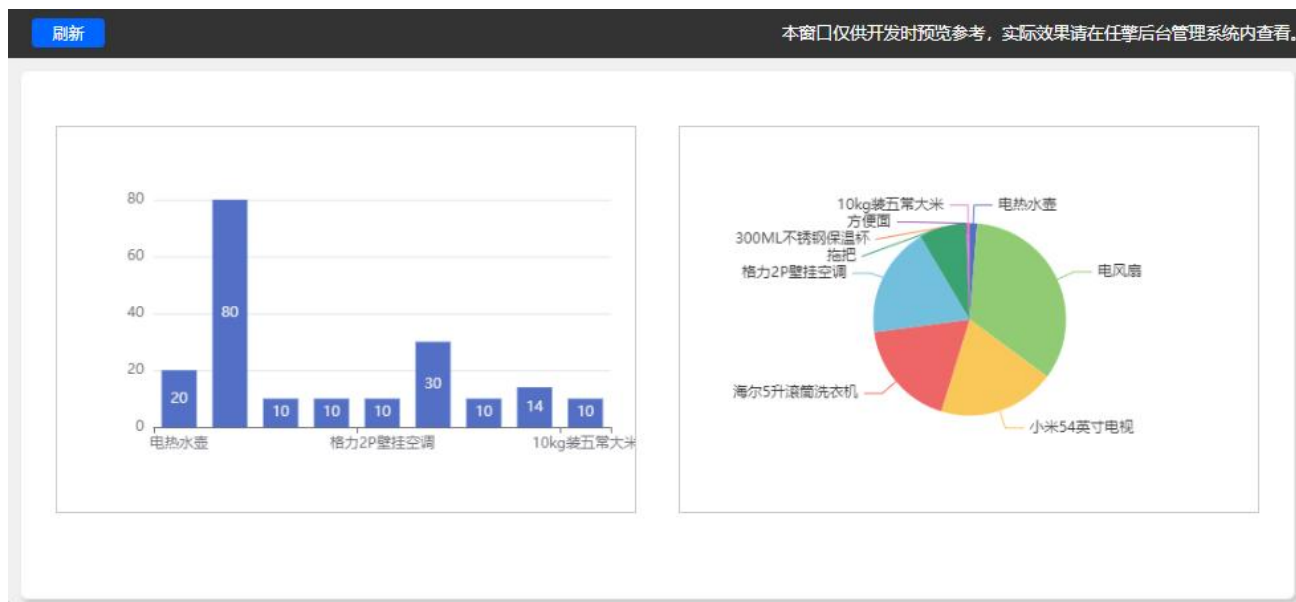
## 6.6.2 拖拽和设置饼状图

从组件库的统计图表中拖拽一个饼状图组件到弹性栅格的右侧面板中，然后点击属性设置里的数据查询设置，在打开的对话框中，首先将查询数据方式设置为“后台服务”，然后在下面选择前面创建的统计产品销售情况的后台服务：demo\_sm/service/product/statByType.tbs，然后点确定，即可完成数据查询设置。

然后在这个饼状图组件的属性里，将分类数据字段设置为：product\_name，表示按产品名

称进行分组显示。

然后再在属性设置里将“数据值字段名”设置为 amount，然后保存页面，然后点击预览按钮，即可查看该页面的运行效果，如下图所示：



## 6.7 实现销售情况按月统计

接下来我们再实现一个稍微复杂一点的统计图表，就是统计最近 30 天内的销售情况。

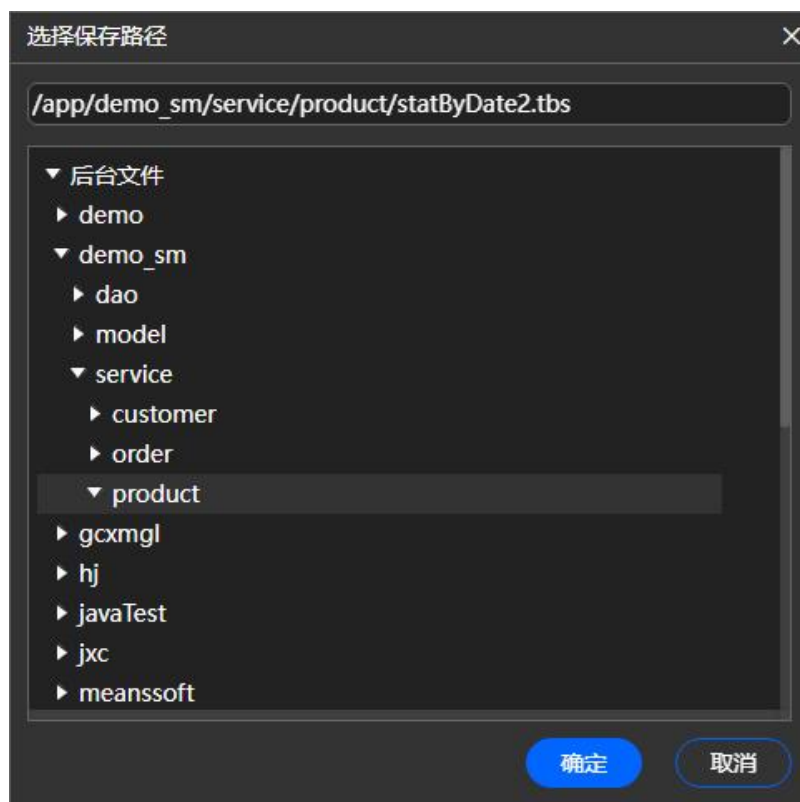
### 6.7.1 创建按日期统计销售情况的后台服务

我们前面创建过一个按日期统计产品分类销售情况的后台服务，这个服务里是按日期和产品分类这两个字段进行数据分组统计的，现在我们只需要按日期分组，则可以将上面这个服务另存为一个新的服务，然后修改里面的统计字段和分组条件即可，操作方法如下：

打开前面创建的按日期统计产品分类销售情况的后台服务：

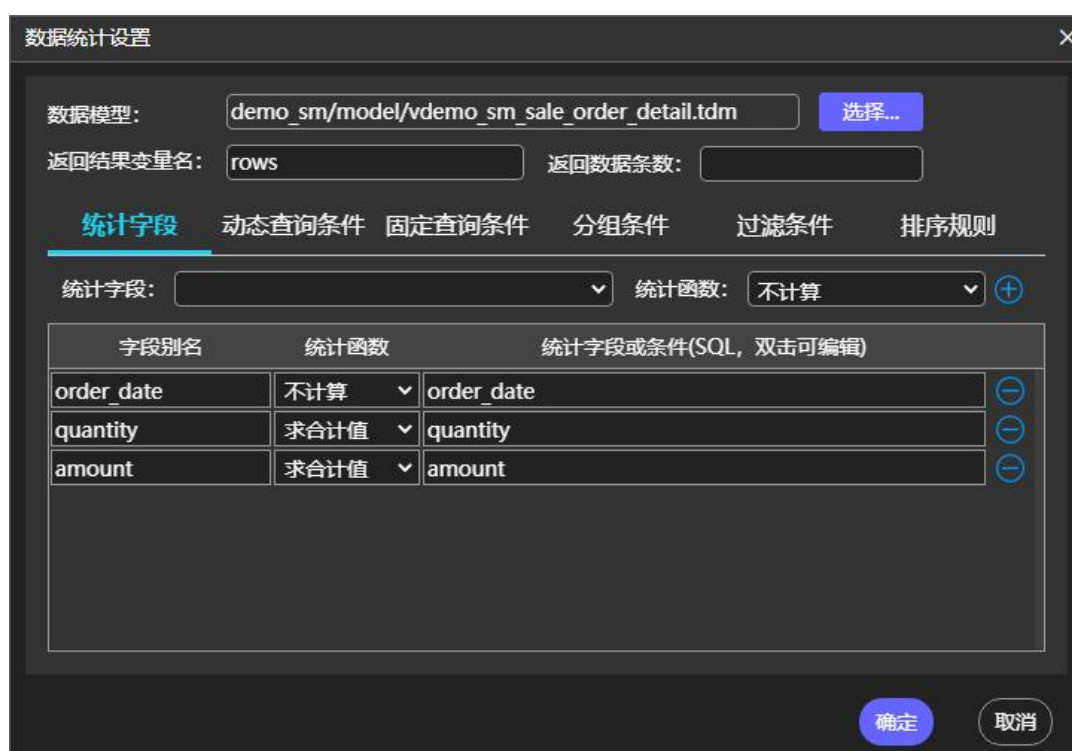
demo\_sm/service/product/statByDate.tbs

点击工具栏上的另存为按钮，然后在弹出的对话框中，将文件名改为：statByDate2.tbs：



然后点确定按钮，完成该后台服务的另存。

然后点数据统计语句里的设置按钮，将统计字段里的 product\_type\_name 字段删除：



然后点击“动态查询条件”选项卡，将里面现有的查询条件删除。

然后点击“固定查询条件”选项卡，在此设置以下查询条件：

`status=1 and datediff(now(), order_date)<=30`

表示仅查询近 30 天内审核通过的销售订单情况。



然后点击“分组条件”选项卡，将里面的 product\_type\_name 字段删除：



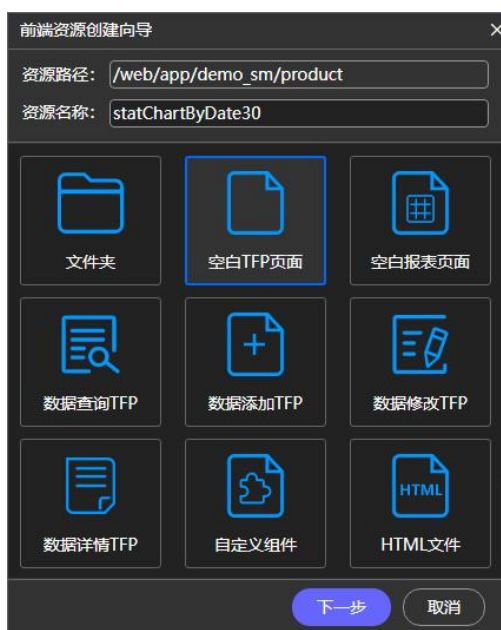
然后再次点击“排序规则”选项卡，将 product\_type\_sort\_num 字段删除：



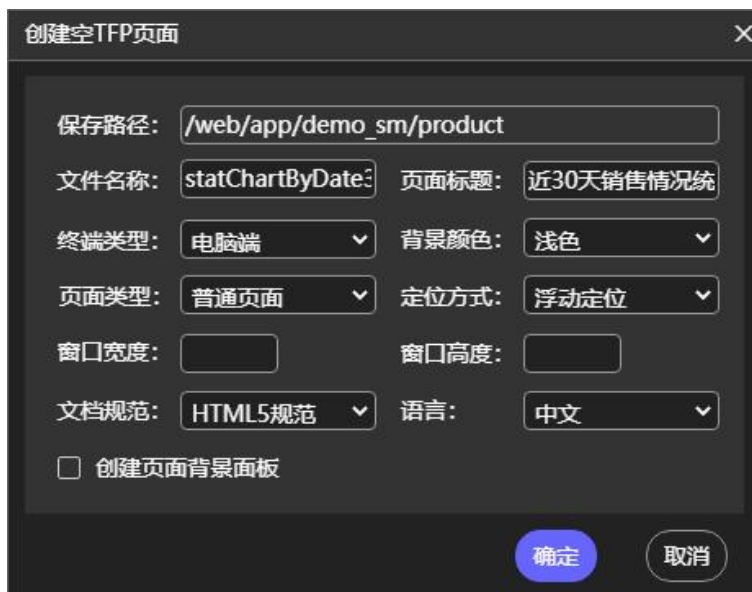
然后点击确定按钮，再点击保存，即可完成该后台服务的修改。

## 6.7.2 创建按日期统计销售情况统计报表页面

在销售管理示例项目的前端文件中，点击 product 文件夹右边的加号，打开如下图所示的“前端资源创建向导”对话框：



在该对话框中选中“空白 TFP 页面”，将资源名称设置为 statChartByDate30，然后点下一步，打开如下对话框：



创建空TFP页面对话框，包含以下配置项：

- 保存路径: /web/app/demo\_sm/product
- 文件名称: statChartByDate30
- 页面标题: 近30天销售情况统计
- 终端类型: 电脑端
- 背景颜色: 浅色
- 页面类型: 普通页面
- 定位方式: 浮动定位
- 窗口宽度: (输入框)
- 窗口高度: (输入框)
- 文档规范: HTML5规范
- 语言: 中文
- ☐ 创建页面背景面板

底部有确定和取消按钮。

在此，将页面标题设置为：近 30 天销售情况统计，然后点确定按钮即可创建一个空白的 TFP 页面。

从容器组件中拖拽一个布局组件到页面中，弹出如下对话框：



布局组件配置对话框，包含以下配置项：

- 顶部面板
  - ☒ 默认显示
  - ☒ 允许折叠
  - 高度: 350 px
- 左侧面板
  - ☐ 默认显示
  - ☐ 允许折叠
  - 宽度: 240 px
- 右侧面板
  - ☐ 默认显示
  - ☐ 允许折叠
  - 宽度: 240 px
- 主面板
- 底部面板
  - ☐ 默认显示
  - ☐ 允许折叠
  - 高度: 150 px

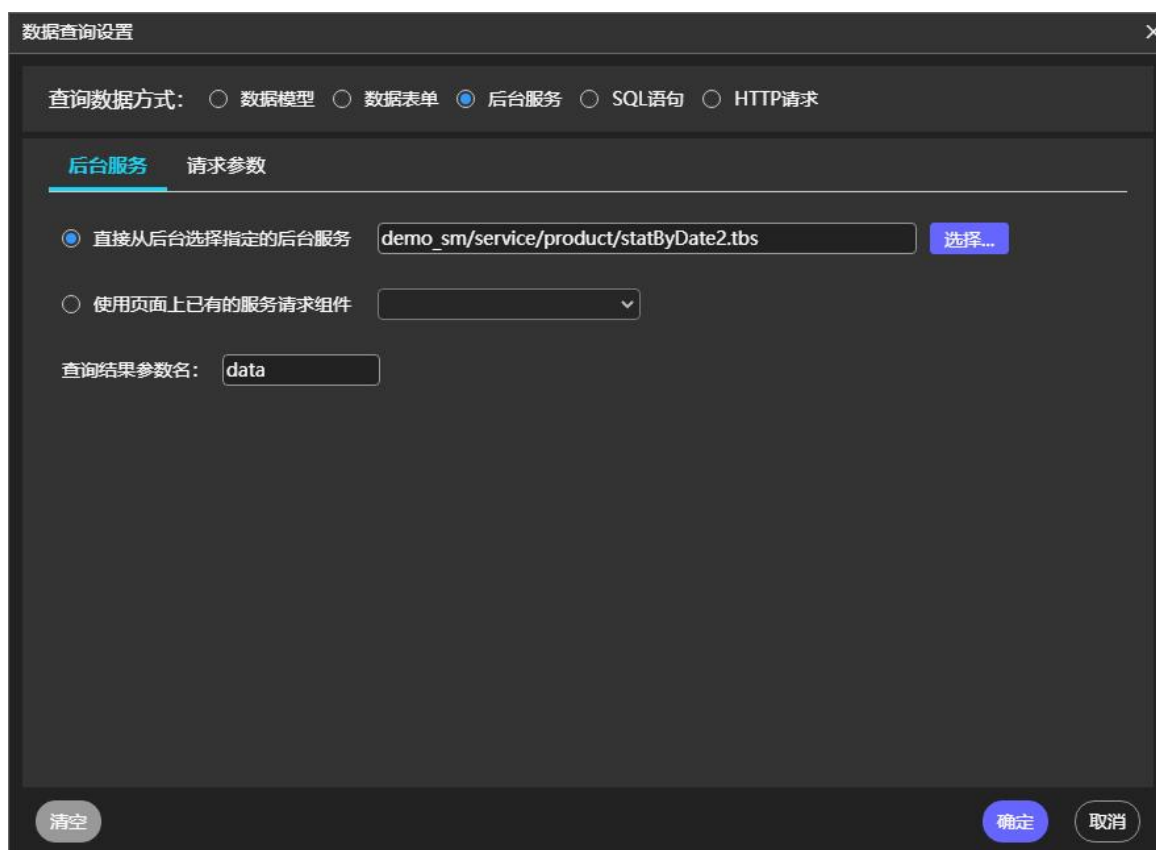
底部有确定和取消按钮。

在此，选择仅显示顶部面板和主面板，其他面板不显示，将顶部面板的高度设置为 450px，然后点确定。



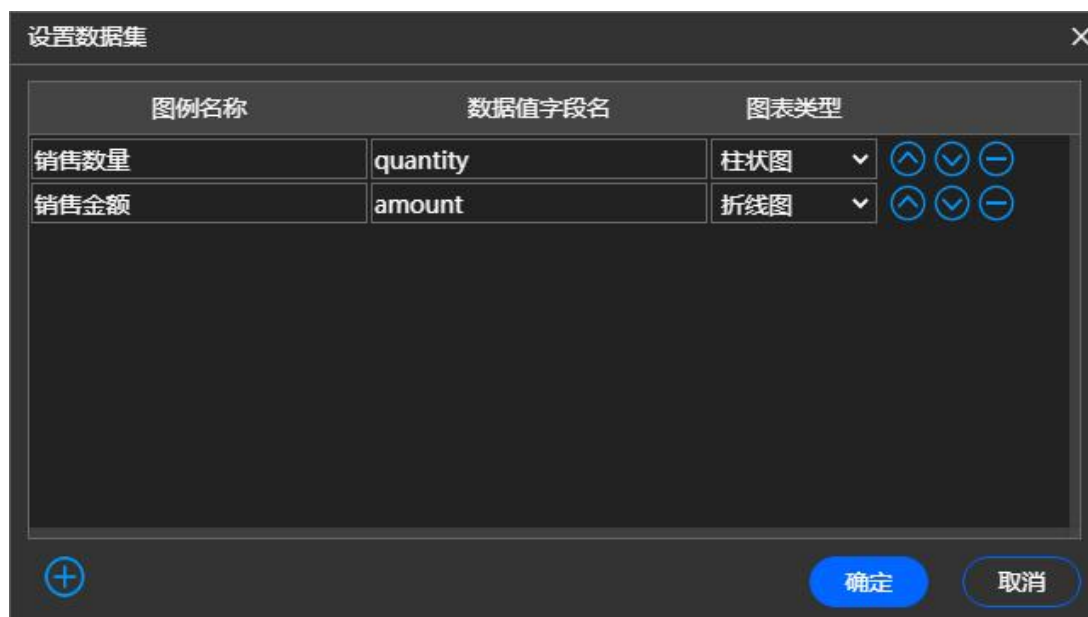
然后从组件库的统计图表里, 拖拽一个线柱结合图组件到上面创建的布局组件的顶部面板里, 并将该组件的宽度设置为 100%。

然后点击属性设置里的数据查询设置, 首先将查询数据方式设置为“后台服务”, 然后在下面选择前面创建的统计 30 天销售情况的后台服务: demo\_sm/service/product/statByDate2.tbs, 然后点确定, 即可完成数据查询设置。



然后在属性里, 将数据分组字段名设置为 order\_date。

然后点击属性设置里的“数据值字段设置”, 在弹出的对话框中, 按下图所示添加和配置两个数据统计字段:



然后点击确定按钮，完成统计图表的设置。

然后再从组件库中拖拽一个数据表格组件到布局组件的主面板中，并将高度设置为 100%。

然后在属性里，点击数据查询设置，首先将查询数据方式设置为“后台服务”，然后在下面选择前面创建的统计 30 天销售情况的后台服务：demo\_sm/service/product/statByDate2.tbs，然后点确定。

然后在点击属性设置里的“数据列设置”，在打开的对话框中，添加以下三列：

| 列名   | 水平对齐方式 | 列宽度 | 数据绑定表达式      | 内容格式 | 计算合计值 |
|------|--------|-----|--------------|------|-------|
| 日期   | 居中对齐   | 120 | {order_date} | 文本   |       |
| 销售数量 | 居中对齐   | 100 | {quantity}   | 整数   | 是     |
| 销售金额 | 右对齐    | 150 | {amount}     | 金额   | 是     |

然后在属性设置的表尾设置里，将显示合计栏开关打开。

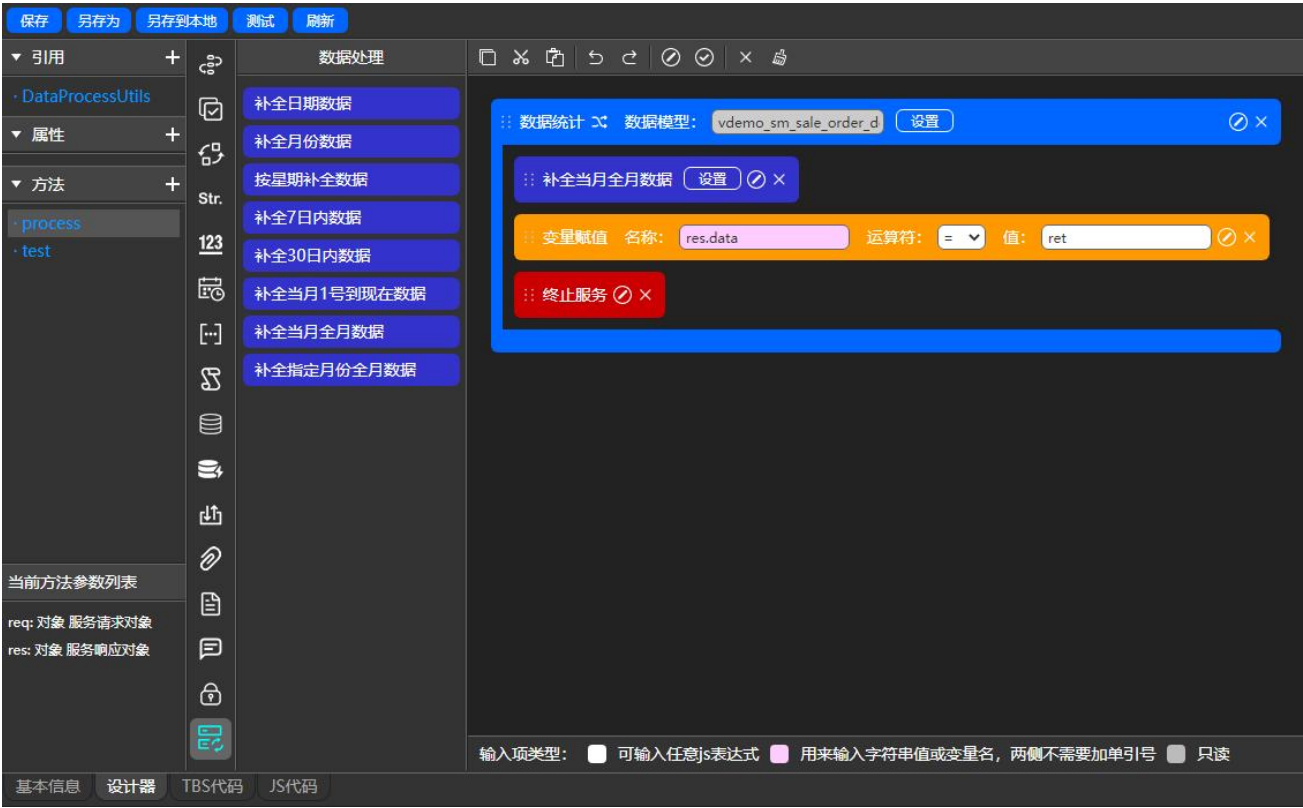
然后保存页面，点预览，即可查看运行效果，界面如下：



### 6.7.3 优化日期分组显示

从上面的统计图可以看到，如果某一天没有销售数据，则统计图和统计表里都不会显示这一天的数据，造成统计图表数据不连续，为此，TaskBuilder 在后台服务的语句库中，专门提供了一套数据处理有关的语句，可以实现统计数据的补全，例如上面的统计页面，如果我们想查看当月每一天的统计数据，我们可以按以下方法操作：

打开上面创建的按日期统计销售情况的后台服务：statByDate2.tbs，从语句库的数据处理中拖拽一个“补全当月全月数据”的语句，放到变量赋值语句上面，如下图所示：



然后点击该语句的设置按钮，打开如下对话框，在此按下图设置相关参数：

参数设置

语句名称：

补全当月全月数据

| 参数名    | 必填 | 参数类型 | 参数值             |
|--------|----|------|-----------------|
| 数据集变量名 | 是  | 字符串值 | rows            |
| 日期字段名  | 是  | 字符串值 | order_date      |
| 数据值字段名 | 是  | 字符串值 | quantity,amount |
| 返回参数名  | 是  | 字符串值 | ret             |

如果参数类型是表达式，则可输入任意js表达式，这时如果输入字符串，两边需要加单引号；如果参数类型是字符串值，则直接输入字符串即可，两边不需要加单引号。

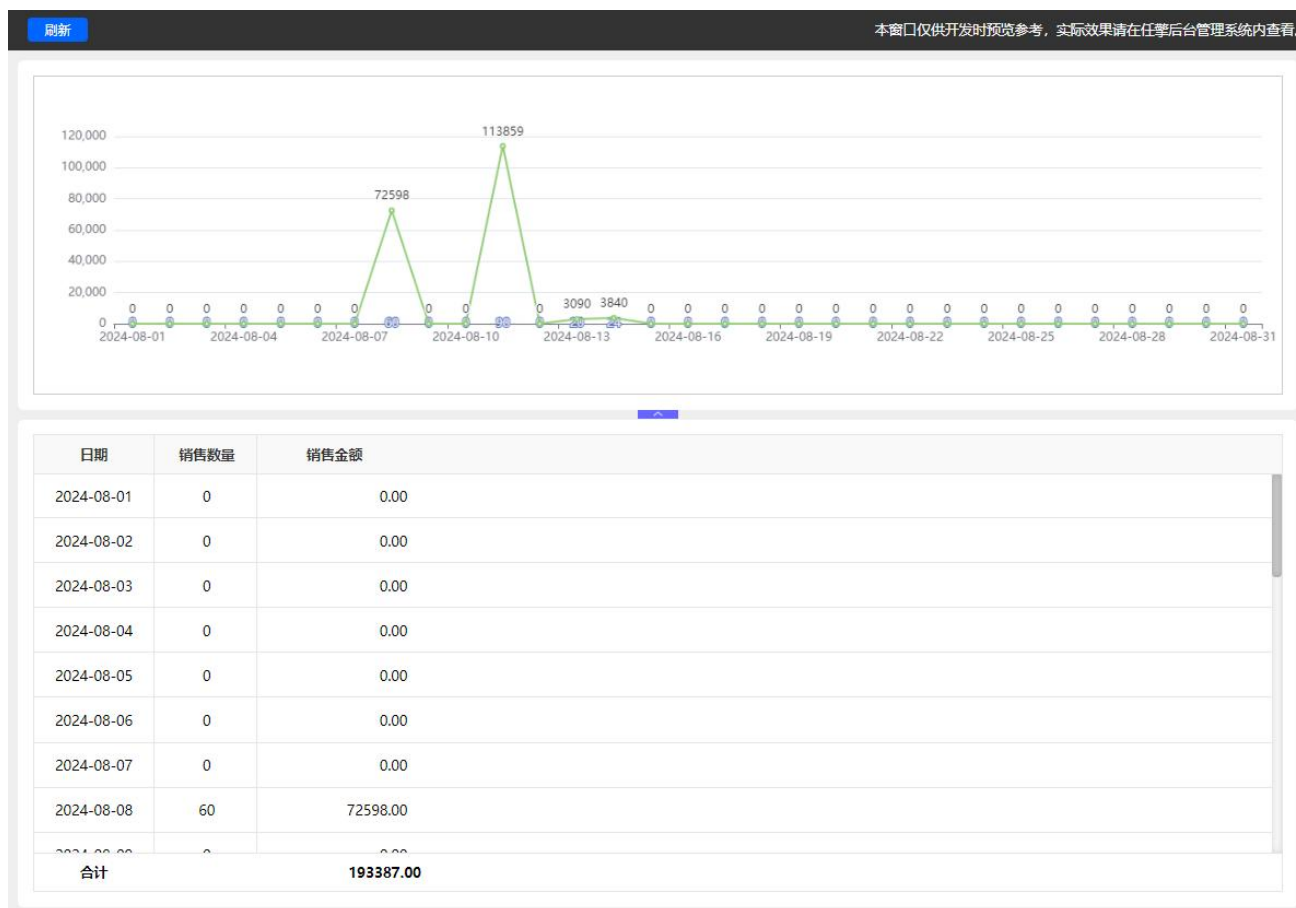
确定取消

上图中的参数说明如下：

- **数据集变量名：**用来设置存储统计数据查询结果的数组变量名，这里设置为 rows；

- **日期字段名：**用来设置存储日期值（必须是 yyyy-MM-dd 格式）的字段名称，这里设置为 order\_date；
- **数据值字段名：**用来设置包含统计数据的字段名，多个字段用英文逗号分隔，这里设置为 quantity,amount；
- **返回参数名：**用来设置处理完后返回的结果参数名；

设置好后，再将下面的变量赋值语句里的变量值修改为 ret，然后点保存，再打开上面的统计页面，点预览，就会发现，统计图和统计表里显示了当月每一天的数据，如下图：



整个销售统计模块开发完成后，就可以在后台管理系统的系统维护的应用管理中，添加对应的应用信息，每个统计页面都可以作为一个单独的应用添加，具体怎么操作可以参考前面的介绍，这里就不再复述。

各位读者如果在使用 TaskBuilder 的过程中有任何问题, 可以加入我们的 QQ 或微信技术支持群进行咨询, QQ 群直接扫码即可, 微信群因为超过人数限制, 需要先加熹哥微信, 然后由熹哥邀请才能进群, 群里有很多开发者分享和交流。



QQ 群: 870548663



熹哥微信

在任擎安装包的 doc 目录内提供了 TaskBuilder 有关的最新文档, 欢迎大家查阅。

### 其他学习资料:

使用指南: <http://www.taskbuilder.org/app/website/guide.tfp>

API 文档: <http://www.taskbuilder.org/app/website/api.tfp>

B 站视频教程: <https://space.bilibili.com/1143074353>