Astro 轻应用

最佳实践

文档版本	01
发布日期	2024-12-31

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 AstroZero 最佳实践汇总

本文汇总了基于Astro轻应用（AstroZero）常见应用场景的操作实践，为每个实践提供详细的方案描述和操作指导，帮助用户深入了解AstroZero的各个功能。

表 1-1 AstroZero 最佳实践一览表

最佳实践	说明
通过AstroZero开发设备维修管理应用	本实践以IoT领域电梯设备运维管理和维修的应用场景为主线，由浅入深的向您介绍如何在AstroZero中开发该应用。
使用AstroZero将客户与订单数据关联并同步修改	在某些订单系统中，通常需要将客户信息和订单数据进行关联，用于处理订单、扣减库存等。本实践主要向您介绍如何将两个对象进行关联，实现对象数据的同步修改功能。
使用AstroZero在前端表格中增删改对象数据	本实践主要向您介绍如何通过增加一个工具栏，在前端页面实现对象数据的增加、删除和修改。
通过AstroZero中的脚本实现表单的提交限制功能	在开发前端页面时，可以在脚本中为表单添加一些提交限制，来提升用户体验和数据的安全。本实践向您介绍，如何在脚本中定义一个延迟时间，在规定的时间内提交表单时，提示“Submission failed: Not PortalUser!”；超出规定的时间，则提示“Submission failed: Submitted too late”。
通过AstroZero中的脚本实现表格数据的增加和删除	本实践向您介绍如何通过脚本，实现表格数据的增加和删除。
通过AstroZero流程模板创建出差审批电子流	AstroZero低代码平台基于业界BPMN 2.0标准，实现了自己的业务流程管理系统，即工作流。本实践通过创建一个出差审批应用，帮助您快速熟悉AstroZero中的工作流。
为AstroZero标准页面中表格的数据增加链接	在标准页面中，支持为表格中的数据增加超链接，来提升用户体验和数据交互的便捷性。本实践向您介绍，如何在表格的webName列中，将鼠标移动至WEB A上在页面的左下角可查看到对应的链接地址，单击会跳转到对应的页面。

最佳实践	说明
为AstroZero标准页面中的表格增加求和等计算能力	在标准页面中，支持为表格增加求和、求积等计算能力，来提升数据的处理效率。本实践主要向您介绍，如何将表格中“商品花费”列的值设置为 “商品数*价格+其他成本”。
为AstroZero调查问卷应用新增调查项	调查问卷页面中的问卷项由对象模型定义，如果需要添加或修改调查项，需要先修改对象模型“问卷记录表”。本实践向您介绍如何通过对象模型，为调查问卷应用新增调查项“您最常使用的功能或者您最感兴趣功能有哪些？”。
在AstroZero标准页面的表格中显示图片	在表格中显示图片可增强信息的表达效果，信息更直观、生动和易于理解。本实践向您介绍，如何在标准页面中，通过自定义列的显示类型，将图片显示在表格中。
使用AstroZero自定义组件在页面中的属性	组件预置的属性不能满足您的业务需求时，支持为组件自定义属性。本实践向您介绍，如何为组件widget_demo_property自定义Text Property、Checkbox Property和Select Property三个属性。
使用AstroZero为组件配置中英文语言属性	为组件配置多语言属性，实现组件在不同语种环境下都可正常显示。在AstroZero中对组件进行国际化配置，主要是修改国际化资源文件（i18n）。本实践向您介绍，如何为组件widget_demo_i18n配置中文和英文两种语言属性。
使用AstroZero创建高级页面适配多终端显示	当用户开发的高级页面应用于多种设备时，如何才能保证在不同大小的设备上，能够呈现同样的网页？为此，AstroZero提供了高级页面的电脑端和移动端两种终端视图、流式布局的响应式布局，并为绝对布局提供“拉伸”功能辅助自适应。 本实践以开发一个满足响应式布局的商品列表组件为例，向您介绍如何适配多终端。
使用AstroZero开发高级页面时如何引用第三方库	库是支撑高级页面组件运行的第三方依赖，如果缺少相应的库，则高级页面组件不能正常运行。本实践向您介绍，如何引入第三方库MintUI，来降低组件开发复杂度、丰富组件的功能。
在AstroZero高级页面中使用轮播组件实现图片展示和URL跳转	高级页面中的轮播组件主要用于多个图片的自动循环切换，您也可以为图片添加超链接，即单击图片，跳转到指定的网站。本实践向您介绍，如何为轮播组件中图片添加一个超链接，跳转到AstroZero帮助文档。
在AstroZero高级页面中使用花瓣图展示订单数据	高级页面中组件展示的数据除了系统预置的静态数据之外，还支持动态数据，即通过调用脚本、服务编排或对象等接口动态生成的数据。本实践向您介绍如何在高级页面中，通过玫瑰花饼图展示订单数据。
通过AstroZero中的连接器上传并识别身份证图片	AstroZero封装了不同类型的连接器用于对接其他服务，对接后即可在应用中使用该服务。本实践向您介绍如何通过OCR连接器，实现身份证信息的识别和存储能力。

最佳实践	说明
通过AstroZero中的连接器实现文件上传功能	AstroZero封装了不同类型的连接器用于对接其他服务，对接后即可在应用中使用该服务。本实践向您介绍如何通过OBS连接器，将前端页面中上传的文件存储到OBS桶中。
将AstroZero中的应用发布成WeLink轻应用	在AstroZero中绑定WeLink后，可将AstroZero中开发的应用快速发布到WeLink中，实现企业业务的高效率、低成本创新。本实践向您介绍如何把AstroZero上开发的应用发布成WeLink（蓝标）轻应用。
将AstroZero中的应用发布成WeLink We码应用	We码是将前端的静态资源放入WeLink，都是本地访问适用于对页面加载速度要求比较高的场景。本实践向您介绍如何把AstroZero上开发的应用发布成We码应用。
将AstroZero中的应用发布到微信小程序	本实践向您介绍如何将AstroZero上开发的应用发布到微信小程序，包括微信公众平台小程序注册、小程序登录鉴权、发布配置和验证小程序是否发布成功。
通过AstroZero开发业务用户登录页	AstroZero为每个应用预置了一个默认的登录页，业务用户可通过默认的登录页登录应用。本实践向您介绍，如何为应用自定义一个登录页。

2 通过 AstroZero 开发设备维修管理应用

2.1 注册登录及环境说明

2.1.1 注册账号及开通 AstroZero 服务

注册华为账号并实名认证

如果您已有一个华为账号，请跳到**为账户充值**。如果您还没有华为账号，请参考以下步骤创建。

步骤1 进入**华为云**官网，单击页面右上角的“注册”。

步骤2 参考**注册华为账号并开通华为云**中操作，完成注册。

步骤3 注册后参考**个人账号如何完成实名认证**或**企业账号如何完成实名认证**中操作，完成个人或企业账号实名认证。

----结束

为账户充值

实名认证后，请为您的账户进行充值。AstroZero提供了免费版、标准版、专业版和专享版四种规格套餐，供您选择。为了更好的体验本示例，建议您购买专业版或专享版，否则**打包发布**应用到运行环境功能，将不可执行（免费版不提供运行环境）。

- AstroZero规格差异说明，请参见**产品规格差异**。
- AstroZero价格的详细介绍，请参见**Astro轻应用 计费说明**。
- 如何进行账户充值，请参见**账户充值**。

购买 AstroZero 实例

在使用AstroZero前，您需要创建一个AstroZero实例。AstroZero实例是一个独立的资源空间，所有的操作都是在实例内进行，不同实例间的资源相互隔离。购买AstroZero实例的详情操作，请参见**购买AstroZero商用实例**。

说明

AstroZero提供了新版和经典版两套开发环境，本入门以经典版开发环境为例进行介绍。新版开发环境的介绍，请参见**初识新版设计器**。

2.1.2 进入经典版开发环境

什么是开发环境

开发环境是开发者专门用于开发应用的环境，用户可根据需求自行开发配置各种逻辑模块化元素，从而创建各类应用。关于AstroZero中环境的更多介绍，请参见**基本概念**。

进入经典版开发环境

步骤1 使用华为账号，登录**华为云网站**，在顶部导航栏右侧单击“控制台”，进入华为云控制台。

图 2-1 进入华为云控制台

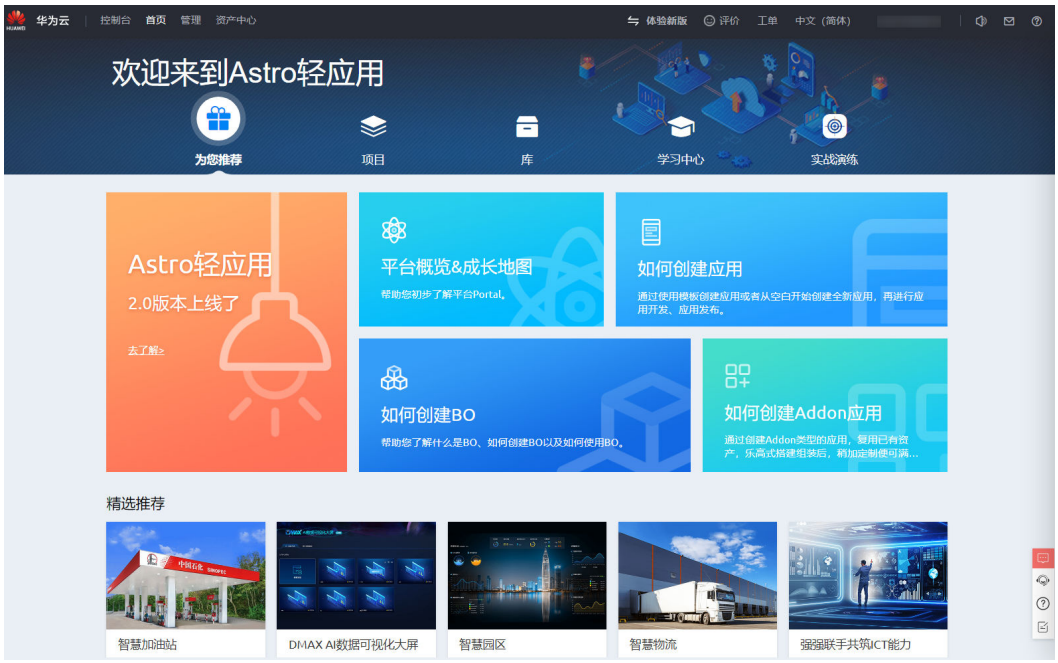
步骤2 在左侧导航栏上方，单击，选择服务所在的区域。

步骤3 单击，在查找框中搜索“Astro轻应用”，单击查找到的结果，进入AstroZero服务控制台。

图 2-2 AstroZero 服务控制台

步骤4 在页面上方，单击“旧版入口”，即可进入经典版开发环境。

图 2-3 经典版开发环境



----结束

2.2 业务场景和流程设计介绍

本文以IoT领域电梯设备运维管理和维修的应用场景为主线，由浅入深的向您介绍如何在AstroZero中开发该应用。

业务场景

设备维修管理系统应用中，包含以下两类用户：

- 系统管理员用户：管理应用的用户，用于新增业务用户，添加用户权限、添加设备信息、管理工单和监控设备。

  说明

  本示例以登录AstroZero开发应用的账号，作为管理员账号。

- 业务用户：使用“设备维修管理系统”应用的用户，分别是客服人员、派单员及维修人员。

设备运维管理和维修场景的业务流程：

1. 系统管理员进行电梯信息的管理和维护，如增加电梯基本信息，修改电梯基本信息等，对业务用户的管理，如新增系统业务用户，并为其分配权限等。
2. 电梯客服人员受理用户投诉，并创建维修单。
3. 派单人员收到客服人员的维修单后，派发给维修工程师。
4. 维修工程师进行现场修理，并在处理完成后关闭维修单。

设备维修管理系统的各个角色涉及的具体业务如下：

图 2-4 业务应用管理功能

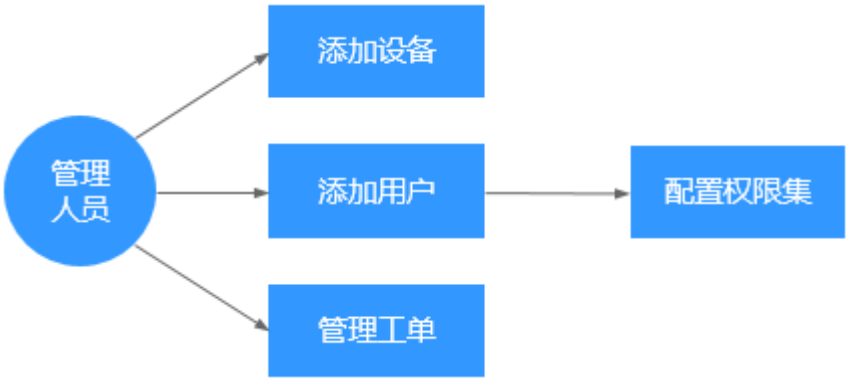
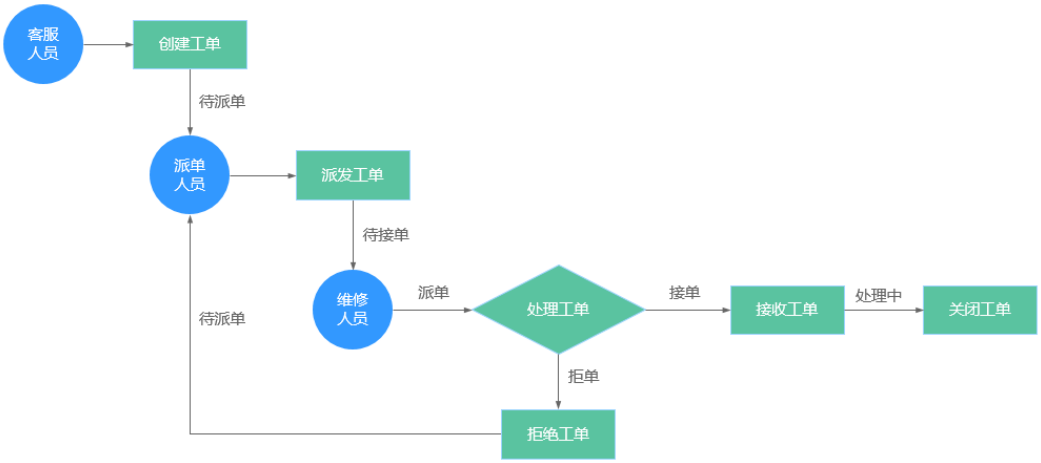


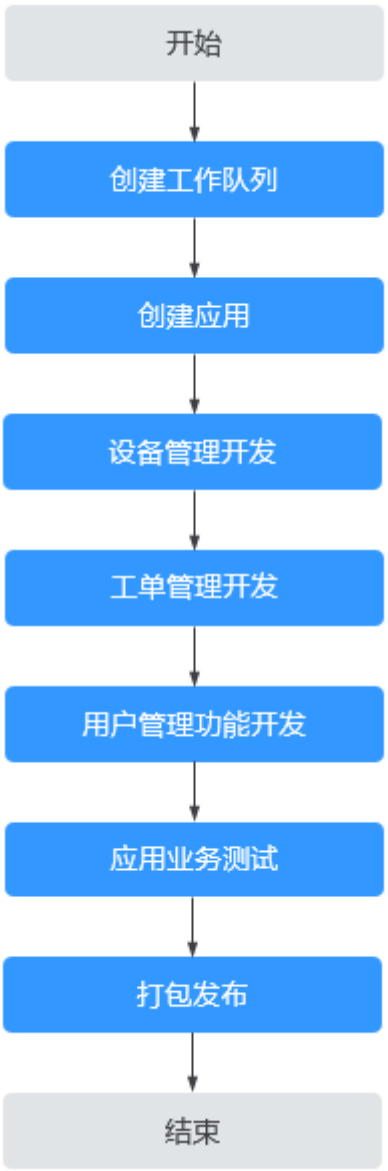
图 2-5 工单流程



开发流程

基于业务场景的应用开发流程，如图2-6所示。

图 2-6 应用开发流程



沿着上述开发线，您将在后续各章节了解到AstroZero的如下能力：

表 2-1 开发任务和平台能力的映射关系

序号	开发任务	预计开发时长	对应的平台能力	学习目标
1	创建工作队列	20分钟	创建工作队列	了解工作队列的含义。

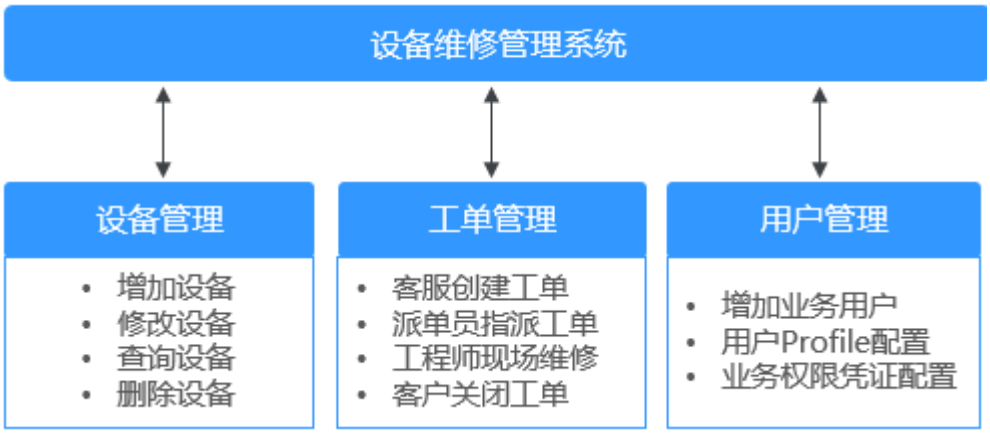
序号	开发任务	预计开发时长	对应的平台能力	学习目标
2	创建应用	1小时	• 创建应用 • 创建高级页面	• 掌握创建应用程序的方法。 • 了解应用程序的目录结构。 • 了解高级页面的相关背景知识和组装方法。 – 上传widget – 使用widget拼装高级页面
3	设备管理开发	4小时	• 创建对象 • 开发脚本 • 开发服务编排 • 组装标准页面 • 组装高级页面	• 了解对象相关背景知识，掌握自定义对象的方法。 • 了解标准页面相关背景知识，掌握使用“表格”组件组装页面、增删改查对象的方法。 • 了解服务编排相关背景知识，掌握开发、测试服务编排的方法。 • 标准页面UI组件：“表单”、“容器”、“输入框”、“下拉框”、“级联选择框”、“标题”、“按钮”的主要配置属性，为“表格”组件添加操作列。 • 页面模型：模型的类型和作用。 • 页面事件：组件事件代码的编写。 • 使用widget组装高级页面。

序号	开发任务	预计开发时长	对应的平台能力	学习目标
4	**工单管理开发**	14小时	• 开发脚本 • 组装复杂的标准页面 • 开发服务编排 • 开发BPM • 定义应用的导航菜单	• 了解脚本相关背景知识，掌握开发、测试脚本的方法。 • 标准页面： – UI组件：了解表单、容器、输入框、下拉框、级联选择框、标题和按钮组件的主要配置属性，为“表格”组件添加操作列。 – UI组件：了解“可折叠容器”的主要配置属性，自定义“表格”组件中某字段的显示内容。 – 事件：动态加载页面内容、组件事件代码的编写。 – 模型：模型的类型和作用。 • BPM：了解BPM、掌握BPM开发流程及图元配置。
5	**用户管理功能开发**	4小时	• 租户和子账号管理 • 权限管理 • 配置权限 • 配置业务凭证	• 了解租户和业务用户。 • 了解平台的权限控制机制：平台操作权限、应用权限、标准对象权限、自定义对象权限平台。
6	**应用业务测试**	30分钟	• 应用预览环境	• 了解应用预览环境。 • 配置应用布局。
7	**打包发布**	10分钟	• 打包 • 发布	• 了解应用的编译发布。 • 了解在运行环境安装应用，了解如何在运行环境中为应用设置自定义登录页面。 • 在其他租户环境安装应用包。

2.3 功能模块分类说明

设备维修管理系统包括三个功能模块，各功能模块分类说明，如图2-7所示。

图 2-7 功能模块说明



本文将以功能模块为单位，逐个开发，带您一步步学习如何使用AstroZero开发一个完整的应用。

2.4 创建工作队列

工作队列是在业务场景中，用来记录一类具有相同权限和任务对象的成员集。创建工作队列是为了给业务用户在业务工单流转过程中，区分不同的权限。

本文中的工作队列用于工单流程中的**业务用户群**，包括客服人员、派单人员和维修人员三个工作队列。

背景信息

工作队列是在业务场景中，用来记录可以受理相同具体业务的用户群体。 这种记录一直保留在队列中，直到用户接受它们并进行处理，或它们被转移到另一个队列。您可以指定每个队列支持的对象集合，以及允许从队列检索记录的用户组（用户、业务用户等）。

创建工作队列主要包含两方面内容：

1. 添加队列的成员。

   队列成员可以是单个用户、公共组、单个角色或带有下级角色的角色，以及业务用户；如果队列的成员是角色，此队列将包含角色中所有的用户。

2. 配置队列支持的对象。

   配置了支持的对象后，涉及特定数据对象的触发器、待审批任务才能进入该队列。

创建工作队列并添加成员

步骤1 使用华为账号，参考进入经典版开发环境中操作，进入AstroZero经典版开发环境。

步骤2 在经典版应用开发环境中，单击“管理”，进入AstroZero经典版开发环境管理中心。

图 2-8 管理页面入口

步骤3 在左侧导航栏中，选择“用户管理 > 工作队列”，单击“新建”。

图 2-9 工作队列入口

步骤4 新建“客服人员”队列。

1. 在新建队列页面，设置队列“标签”为“客服人员”，“名称”为“CustomerService”。

2. 在“队列成员”中，单击“添加”。
3. 设置“成员类型”为“用户”，“成员列表”选择当前租户开发者，再单击“添加”，退出弹窗。

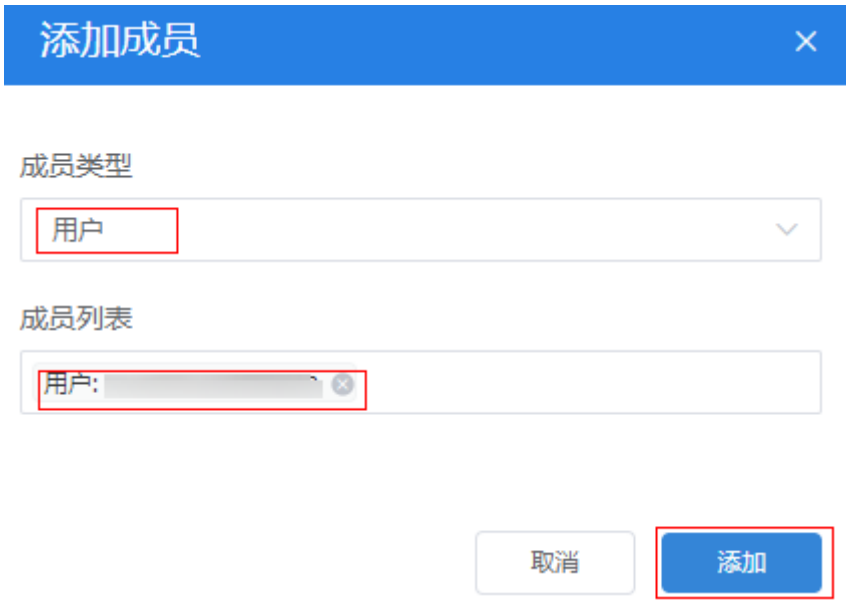
说明

当前还未创建业务用户，因此这里只添加当前开发者账号到队列中，便于后续开发测试。

4. 在新建队列页面，单击“保存”。

图 2-10 添加用户

图 2-11 添加“用户”成员



步骤5 参考以上步骤，创建表2-2中的其他工作队列。

表 2-2 工作队列详情

标签	名称	成员列表
客服人员（上一步已创建）	CustomerService	当前应用开发者、业务用户列表成员中的客服人员。
派单员	Dispatcher	当前应用开发者、业务用户列表成员中的派单人员。
维护人员	FME	当前应用开发者、业务用户列表成员中的维护人员。

----结束

后续操作

因为业务用户是在应用创建完成后才创建的，因此当前还不能将业务用户中的“客服人员”、“派单人员”以及“维护人员”，添加到工作队列中，在应用业务测试章节，新增业务用户之后，则需要将每个业务用户添加到工作队列中。

2.5 创建应用

2.5.1 创建“设备维修管理系统”应用

应用是一个可运行的、实现特定业务功能的业务单元。创建应用是在AstroZero开发项目的第一步，也是端到端构建软件应用的入口。

“设备维修管理系统”应用的主要功能包括电梯设备信息管理、业务用户管理，维修工单的创建/派发和处理调度，以及整个维护情况的监控分析。

创建应用

步骤1 使用华为账号，参考**进入经典版开发环境**中操作，登录AstroZero经典版开发环境。

说明

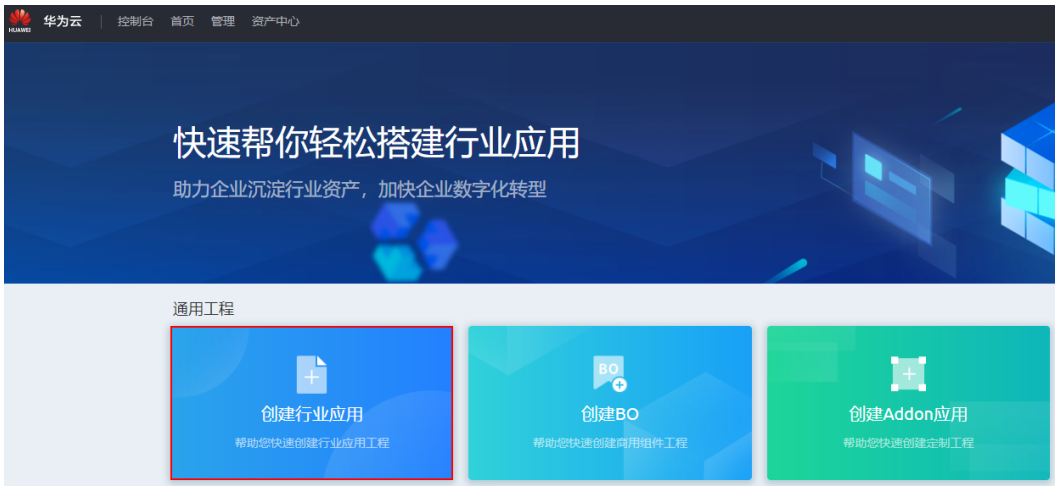
此处的华为账号是指购买AstroZero服务的账号。更多AstroZero中用户的介绍，请参见**AstroZero中的用户**。

步骤2 在AstroZero经典版开发环境首页的“项目”页签下，单击“行业应用”。

图 2-12 创建行业应用

步骤3 在行业应用页面，单击“创建行业应用”。

图 2-13 创建空白行业应用



步骤4 定义命名空间。

为保证后续您的应用发布到应用市场时，不与其他开发者冲突，首次创建应用时，需要先定义本租户的命名空间。

后续您创建的对象、脚本、服务编排、页面等的名称中，系统都会为其增加命名空间前缀。例如，您将命名空间定义为“HW”，则您后续创建的所有对象、脚本、页面等都将以“HW__”开头。

⚠ 注意

- 命名空间一旦设定不能修改，请慎重定义，建议您使用公司或者团队的缩写作为命名空间。
- 本实战中以命名空间“HW__”为例进行说明，实际操作时请替换为您所创建的命名空间。

步骤5 选择“应用”页签，输入应用程序的“标签”和“名称”，单击“创建”。

应用创建完成后，自动进入应用开发页面。

图 2-14 创建应用

表 2-3 应用基本信息

标签	名称
设备维修管理系统	MyApp **说明** • 应用创建后，将自动添加命名空间前缀，实际创建应用名为“HW__MyApp”。 • 应用名称在AstroZero中是应用的唯一标识符，在应用开发过程中，例如接口调用、业务用户登录等场景，会涉及拼接应用名称。 • 在应用开发页面，如果提示“您的租户没有配置默认邮件服务器地址，将会导致在服务编排/BPM/脚本中发送邮件功能不可用”，请直接忽略，本示例不涉及。

应用创建后，将会自动出现在经典版开发环境首页“项目 > 我的应用”下。再次编辑应用时，可以在“我的应用”下，单击应用名，进入应用。

图 2-15 我的应用

----结束

创建应用目录

步骤1 使用华为账号，参考**进入经典版开发环境**中操作，进入AstroZero经典版开发环境。

步骤2 在AstroZero经典版开发环境首页“项目”页签，单击“我的应用”下的“设备维修管理系统”，进入应用。

步骤3 单击“设备维修管理系统”后的，再单击“目录”，在弹窗中输入“Equipment”，单击“保存”，创建一个“Equipment”目录。

图 2-16 创建应用目录

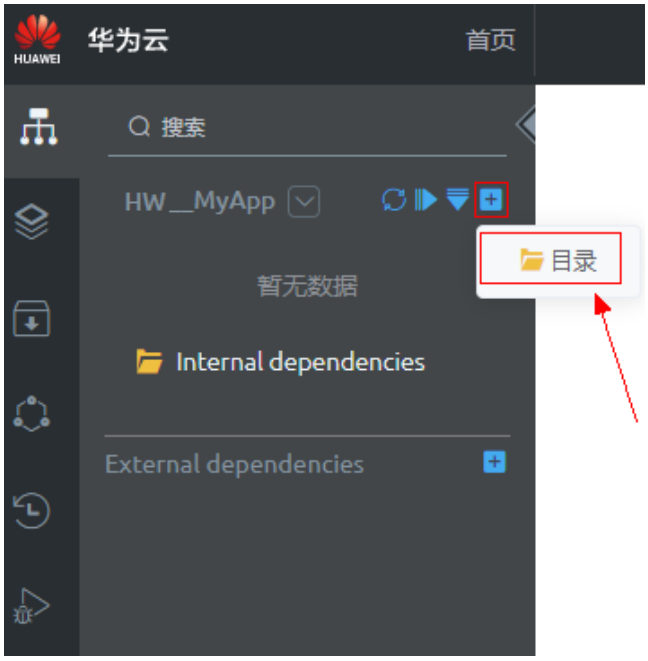
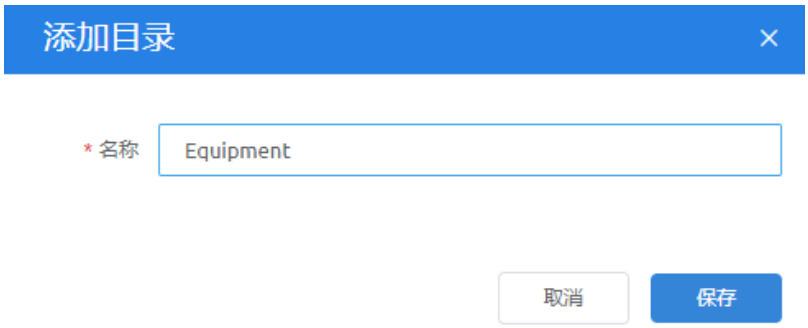


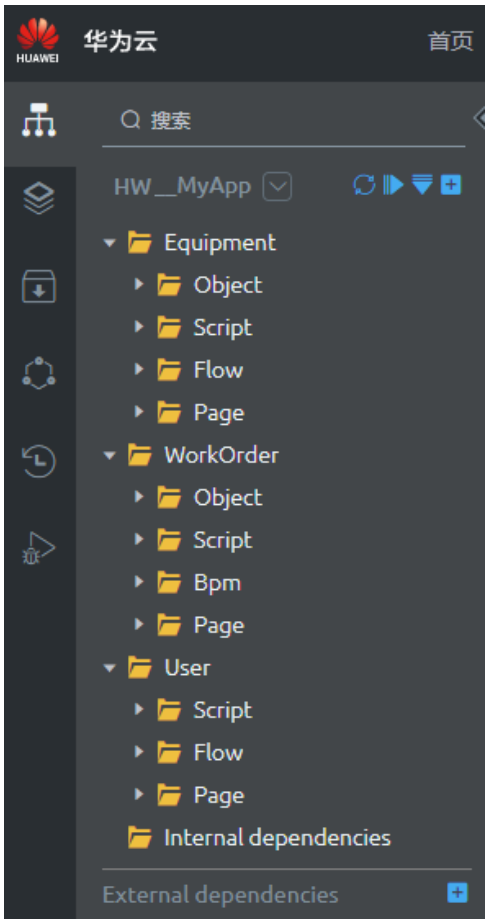
图 2-17 添加目录 Equipment



步骤4 参考上一步，按应用业务的功能模块，再创建WorkOrder和User目录。

步骤5 在Equipment、WorkOrder和User目录下，参考图2-18所示，分别创建子目录Object、Script、Flow、Bpm和Page。

图 2-18 创建应用目录



----结束

2.5.2 关闭 Vue3 框架渲染组件开关

本案例所涉及到的自定义组件是基于Vue2框架开发的，而系统是默认开启Vue3框架渲染组件的，所以您需要手动关掉Vue3框架渲染组件开关，否则拖拽组件到页面时会提示如下报错。

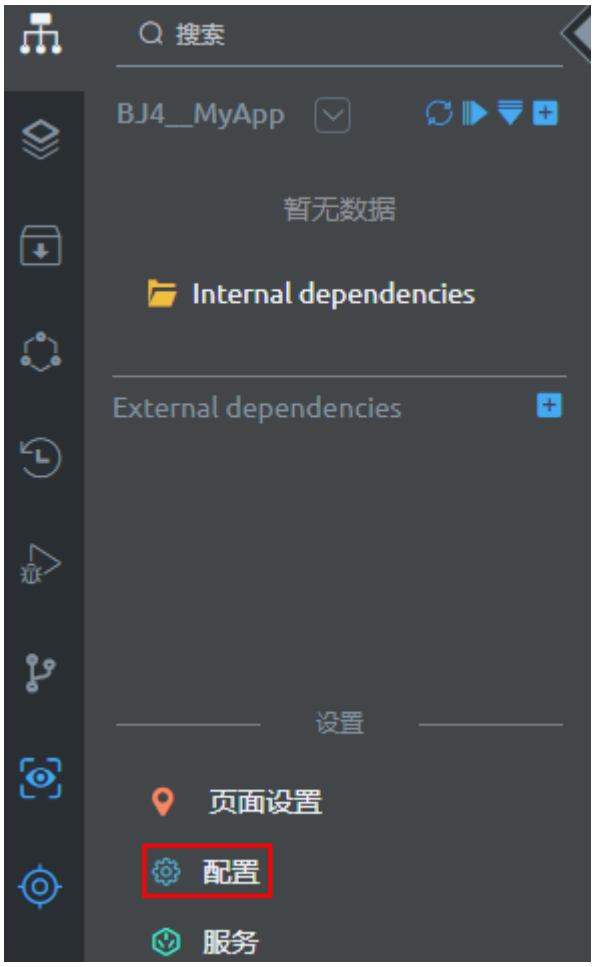
图 2-19 界面报错

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在设备维修管理应用的设计页面，单击下方的“设置 > 配置”。

图 2-20 选择配置



步骤3 在“高级设置”页签中，取消勾选“页面组件的渲染框架由Vue2升级为Vue3”开关。

图 2-21 取消选中

----结束

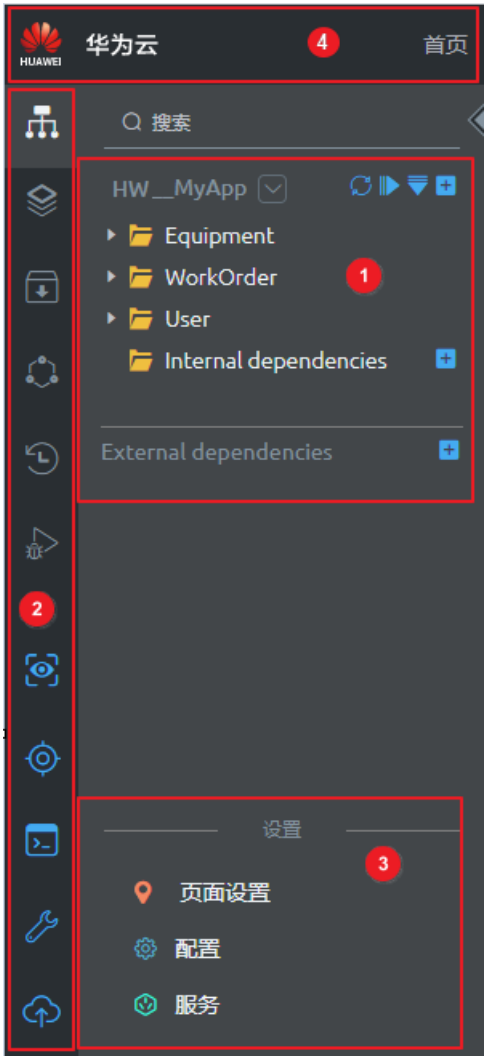
2.5.3 了解应用目录及菜单

一个业务系统通常包括前端页面、后台逻辑和数据库表。因此，AstroZero采用类似的结构管理应用程序。

如图2-22所示，应用管理页面包含三部分（红框1、红框2和红框3）：

- 红框**1**中是当前应用的页面、模型和逻辑。在后续章节中，开发的页面、对象及后台逻辑，推荐按以下规划的方式，将应用资源分别放在对应的目录下：
  - Object：数据对象
  - Script：脚本
  - Flow：服务编排
  - Bpm：BPM
  - Page：前端页面
- 红框**2**中是应用管理功能，例如预览应用、编译打包、发布应用等。
- 红框**3**中是应用设置功能，例如设置应用导航、页面设置等。
- 红框**4**中是AstroZero应用开发环境的导航菜单，单击“首页”可返回开发环境首页。

图 2-22 应用的管理目录



2.5.4 创建“业务用户登录”页面

2.5.4.1 背景与原理

业务应用构建登录页面时，一般情况下是通过使用AstroZero的高级页面能力实现。您可以通过本节认识高级页面，并了解登录页面的开发流程。

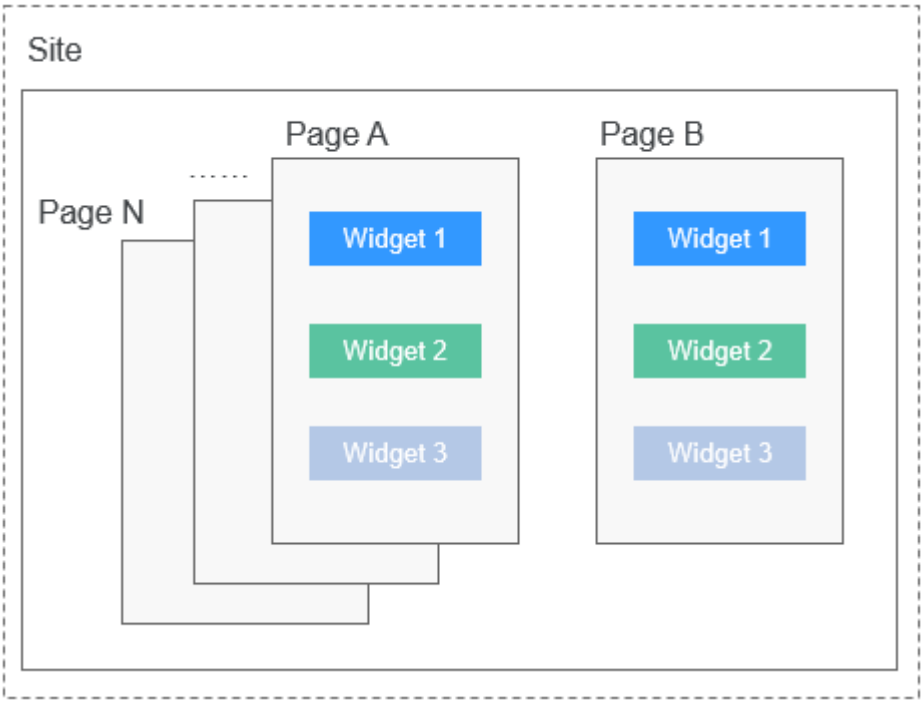
了解高级页面

AstroZero前端页面有标准页面、高级页面和表单三种。本节主要带您了解、学习并使用高级页面。

- 标准页面：对于一般的业务应用系统，其功能主要是针对业务数据的增、删、改、查，前端界面的样式相对简单的页面场景，此时，推荐您使用平台提供的“标准页面”。您可以通过拖、拉、拽页面组件，再加上少量事件代码，即可拼装出所需页面，具体介绍请参见**标准页面**。
- 高级页面：对于一些样式比较复杂的页面，例如网站、电商、园区大屏等，您可以使用平台提供的“高级页面”。

高级页面是由一个或者多个Widget（即组件）拼装而成。如**图2-23**所示，组件是可复用的页面组成元素，一个页面由一个或多个组件拼装而成。如果将一个页面看成拼图游戏的完整图案，那么组件就相当于拼图的每一小块。

图 2-23 页面与 widget（组件）的关系



Widget的运行依赖Library（库），如果缺少相应的Library（库），则Widget不能正常运行。因此在加载widget前，需要先加载必要的Library。

在操作前端页面时，经常会需要调用后台数据，例如即将开发的登录页面，需要获取业务用户信息。这时需要通过桥接器调用后台的服务编排、Script等获取后台数据。因此，在引用widget时，经常需要配置桥接器。

AstroZero高级页面中的组件分为系统预置组件和自定义组件：

- 系统预置组件，可以直接使用。
- 登录组件属于自定义组件。本示例中已经为您提供了开发好的组件包，您只需要上传到站点中即可使用。自定义登录组件的开发方法，请参考**（可选）开发自定义登录组件**。

“业务用户登录”页面开发流程

本示例中，主要通过线下开发一个自定义登录组件，然后上传到高级页面，再进行组件配置，实现应用登录页面功能。登录页面开发流程，如**图2-24**所示。

图 2-24 应用登录页面开发流程

业务用户登录页面即是设备维修管理系统的应用登录界面，大致设想如**图2-25**所示。

图 2-25 应用登录页面

2.5.4.2 （可选）开发自定义登录组件

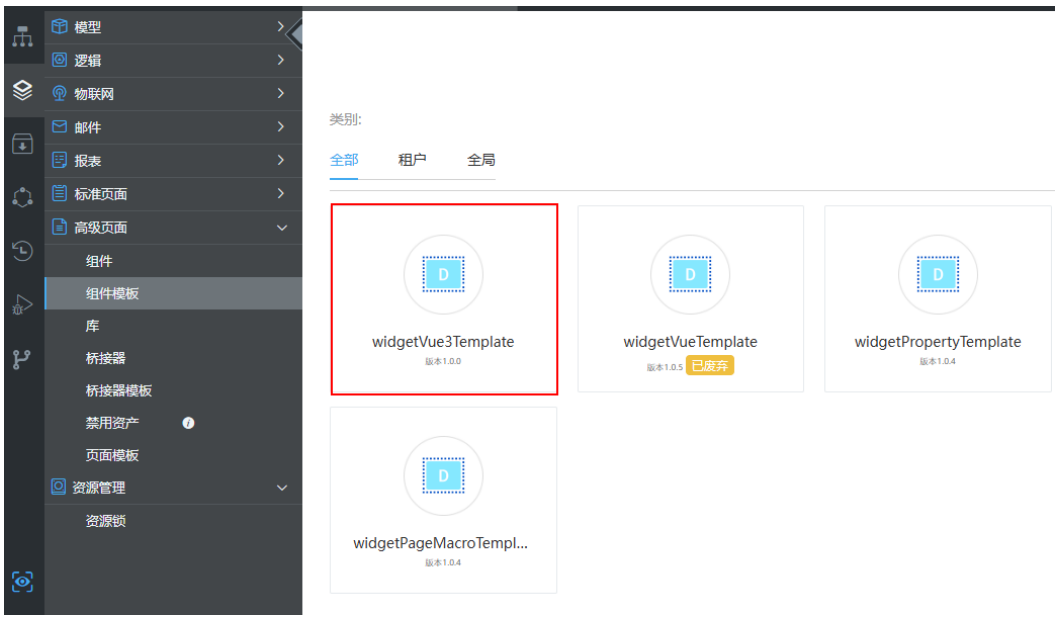
“开发自定义登录组件”步骤可以直接跳过，本示例已为您提供了开发好的自定义登录组件。如果您想要自定义登录组件的开发方法，可参考本章节执行。

自定义登录组件

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

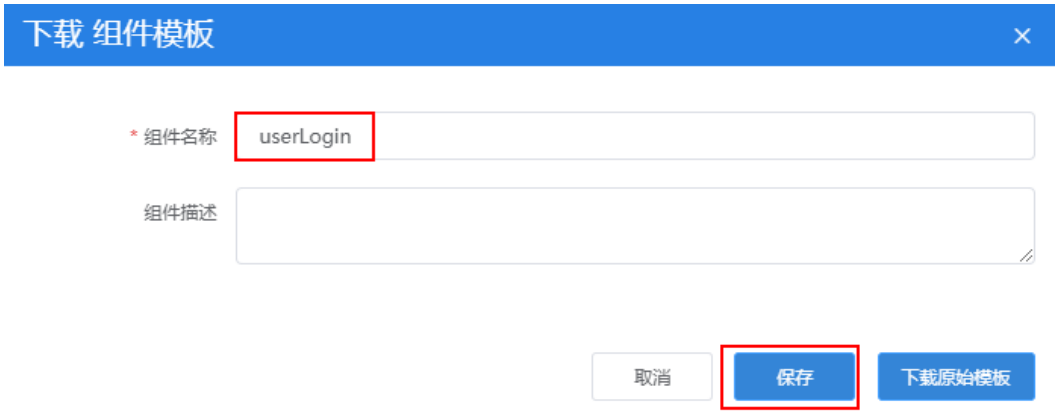
步骤2 在左侧资产下的组件模板中，单击“widgetVue3Template”，再单击“下载”。

图 2-26 下载组件模板



步骤3 在弹出的窗口中，输入组件名称，并单击“保存”，将组件模板保存到本地，并解压。

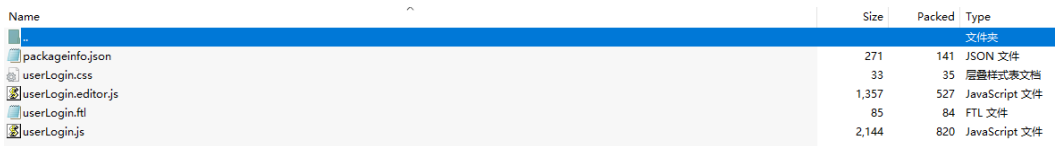
图 2-27 输入组件名称



步骤4 查看解压后的组件目录。

目录结构如图2-28所示，其中userLogin.js文件是写vue业务逻辑的代码，userLogin.ftl用于写html代码，userLogin.css用于写样式代码，userLogin.editor.js以及packageinfo.json是配置文件。

图 2-28 目录结构



步骤5 在本地编辑器中打开文件夹，把userLogin.editor.js文件中的config代码改为如下代码，用于配置桥接器。

```
config: [
            {
               type: 'connectorV2',
               name: 'FlowConnector',
               label: 'Flow Connector',
               model: 'ViewModel'
            },
            {
               type: 'connectorV2',
               name: 'common.GetConnector',
               label: 'View API Get Connector',
               model: 'ViewModel'
            },
            {
               type: 'connectorV2',
               name: 'common.PostConnector',
               label: 'View API Post Connector',
               model: 'ViewModel'
            },
            {
               type: 'connectorV2',
               name: 'common.PutConnector',
               label: 'View API Put Connector',
               model: 'ViewModel'
            },
            {
               type: 'connectorV2',
               name: 'common.DeleteConnector',
               label: 'View API Delete Connector',
               model: 'ViewModel'
            },
         ]
```

步骤6 把packageinfo.json文件中加入如下**加粗内容**。

```
{
   "widgetApi": [
      {
         "name": "userLogin"
      }
   ],
   "widgetDescription": "",
   "authorName": "",
   "localFileBasePath": "",
   "requires": [
      {
         "name": "global_Vue3",
         "version": "3.4.21"
      },
      {
         "name": "global_ElementPlus",
         "version": "2.6.0"
      },
      {
         "name": "global_Vue3I18n",
         "version": "9.10.1"
      },
      {
         "name": "global_Vue3Router",
         "version": "4.3.0"
      }
   ]
}
```

步骤7 将修改后的组件文件压缩成一个zip包。压缩后，即可根据需要上传到AstroZero，供高级页面使用。

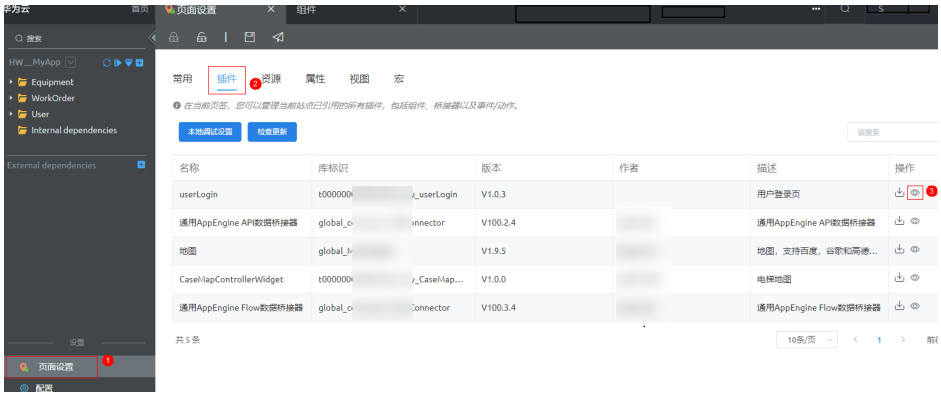
----结束

更新自定义组件

当开发的自定义组件功能有变动，即组件代码发生变动后，需要更新组件版本。更新后，组件所在的页面也会随之生效。

步骤1 在应用中，单击左下方的“页面设置”，再选择“插件”页签，找到需要更新的组件（例如userLogin），单击组件所在行右侧“查看详情”，进入组件详情页。

图 2-29 页面设置下的组件列表



说明

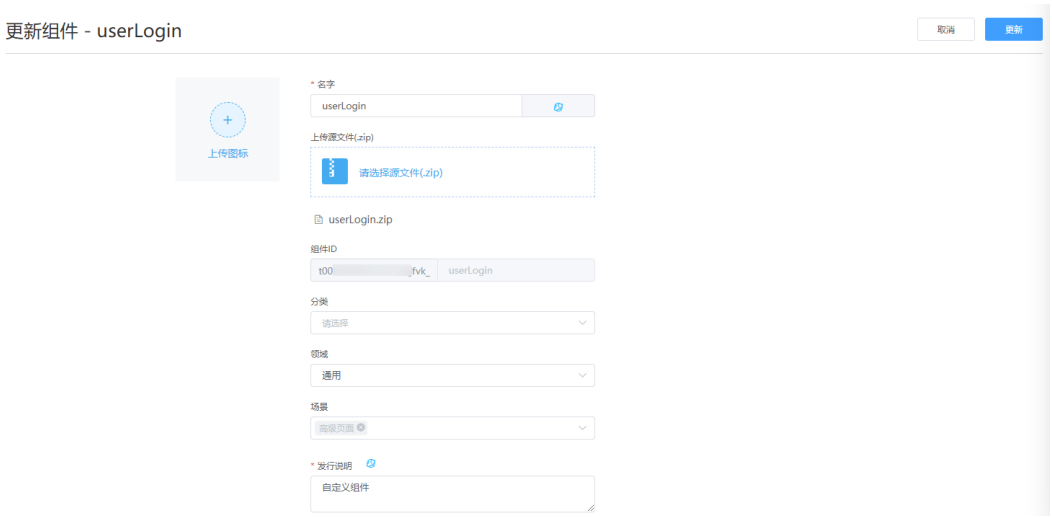
如果页面  图标高亮，则需要先单击  解锁页面。

步骤2 单击“更新”按钮进入组件更新页面。

图 2-30 选择更新按钮

步骤3 单击“请选择源文件(.zip)”，上传本地的组件zip包，再单击“更新”。

图 2-31 上传本地组件包



步骤4 返回“插件”页签，单击刚刚上传组件（userLogin）的升级按钮，然后单击保存升级，最后单击发布即可。

图 2-32 更新组件

----结束

2.5.4.3 上传自定义登录组件

自定义登录组件开发完成，并上传到AstroZero后，才能通过高级页面调用。

您可以直接使用文中提供的已开发完成的登录组件，上传到AstroZero中。也可以自行在AstroZero上，下载自定义组件模板，进行线下开发，然后再上传到AstroZero中。

操作步骤

步骤1 单击**userLogin.zip**，下载自定义登录组件包到本地。

步骤2 使用华为账号，参考**进入经典版开发环境**中操作，进入AstroZero经典版开发环境。

步骤3 在AstroZero经典版开发环境首页“项目”页签的“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤4 在左侧列表中，单击，选择“高级页面 > 组件”，单击“提交新组件”。

步骤5 单击“请选择源文件(.zip)”，选择之前下载的组件包“userLogin.zip”，设置如下参数后，单击“提交”。

图 2-33 上传 userLogin 组件

- 名字：设置新上传组件的名称，本示例设置为userLogin。
- 场景：设置组件的使用场景，后续在创建高级页面中会使用该自定义组件，故此处保持默认即可。
- 发行说明：设置新上传组件的发行说明，通常设置为组件的功能说明，本示例设置为“自定义登录组件”。单击发行说明后的，可对发行说明进行国际化设置。

----结束

2.5.4.4 组装“业务用户登录”页面

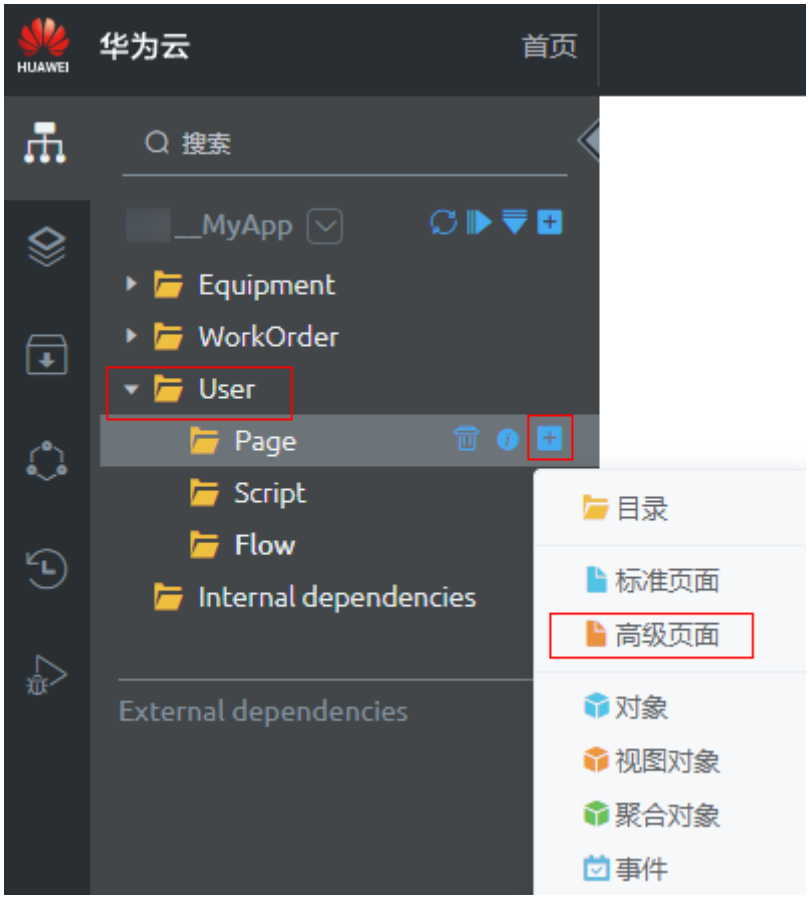
“业务用户登录”页面是一个高级页面，主要是通过引用**上传自定义登录组件**中上传的自定义登录组件，再配置相关参数，来实现登录功能。

操作步骤

步骤1 创建高级页面。

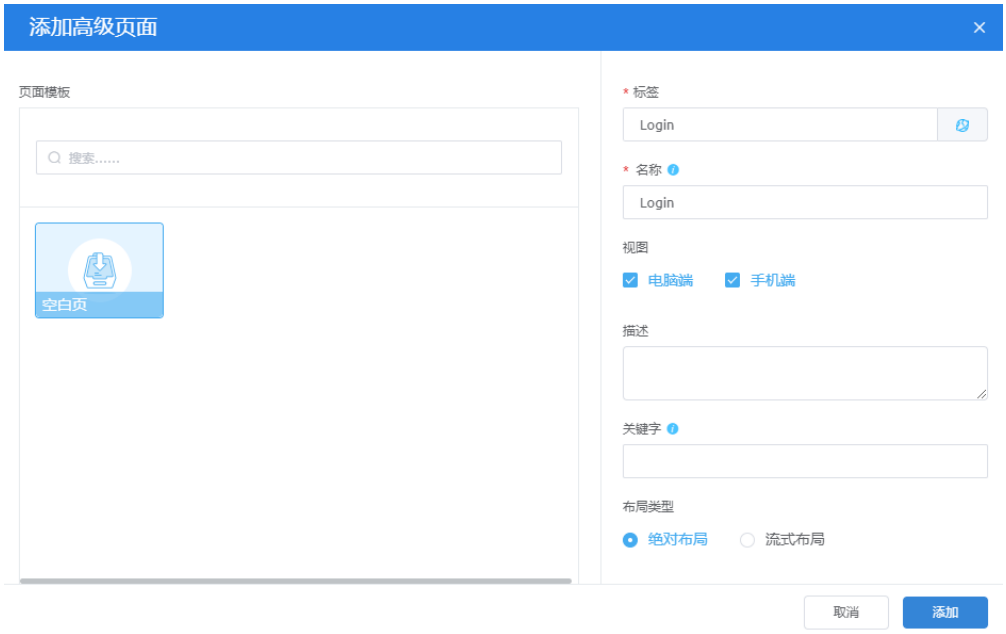
1. 进入AstroZero经典版开发环境首页，在“项目 > 我的应用”中，单击“设备维修管理系统”，进入应用。
2. 单击  ，进入工作目录。
3. 在“User”目录下，将鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“高级页面”。

图 2-34 创建高级页面



4. 设置“标签”和“名称”为“Login”，并选择“绝对布局”，单击“添加”。

图 2-35 添加高级页面



说明

高级页面布局有绝对布局和流式布局两种，页面布局详细介绍请参见**高级页面布局**。在应用中首次添加高级页面时，支持设置视图（电脑端、手机端）。

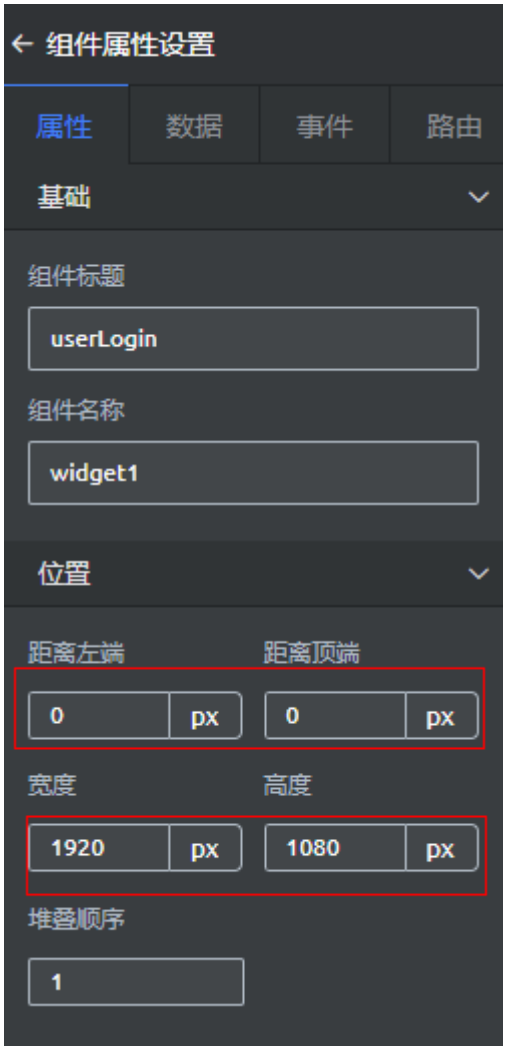
步骤2 拖拽自定义组件“userLogin”到页面。

单击左上角，打开组件列表，单击“全部”。在“自定义”页签，搜索到“userLogin”组件，将其拖进页面编辑区。“userLogin”组件即是在**上传自定义登录组件**中上传的自定义登录组件。

步骤3 设置自定义组件“userLogin”的位置属性。

1. 单击页面右下角的空白处，右侧显示当前视图组件列表。
2. 单击选中“userLogin”组件，会在右侧显示该组件的属性配置面板。
3. 在“位置”中，设置“距离左端”、“距离顶端”为“0”，“宽度”为“1920”，“高度”为“1080”。

图 2-36 设置组件位置



4. 单击页面上方，保存页面修改。
5. 在目录中，将鼠标放在“Login”上，单击，然后选择“设置”，在弹窗中设置页面的拉伸属性，再单击“保存”。

图 2-37 页面设置

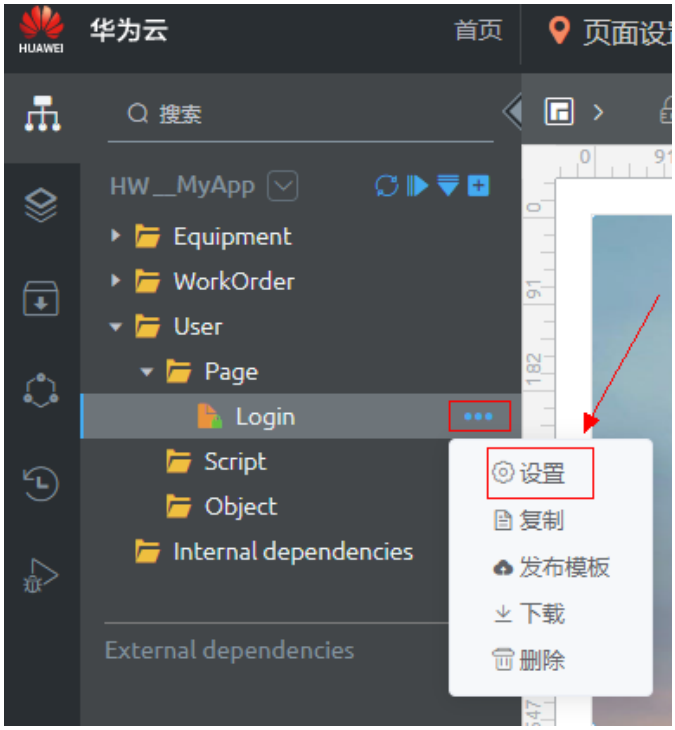
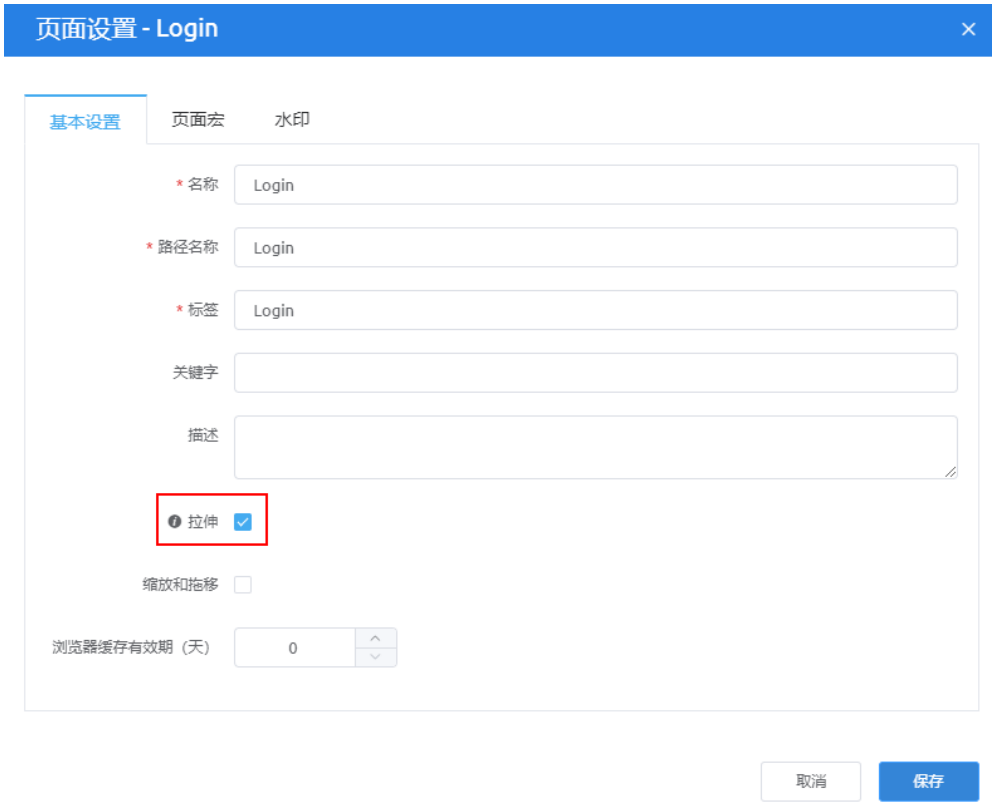


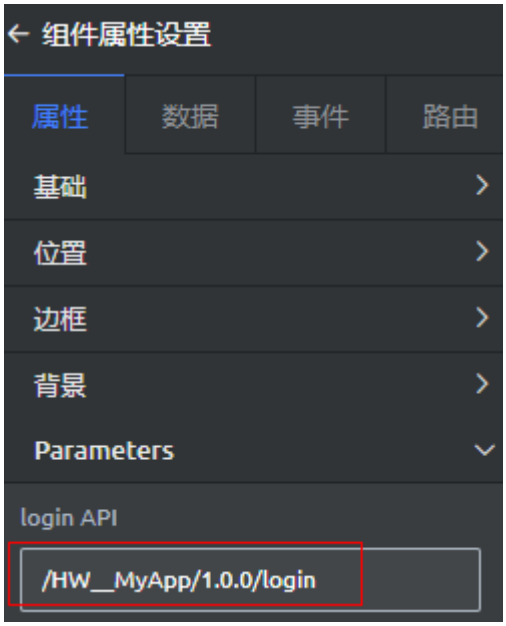
图 2-38 设置页面拉伸



步骤4 设置自定义组件“userLogin”的“数据”中的桥接器。

1. 单击选中“userLogin”组件，在右侧“组件属性设置 >属性”页签最下端的“Parameters”配置项下，设置“login API”为“/***HW***__MyApp/1.0.0/login”。

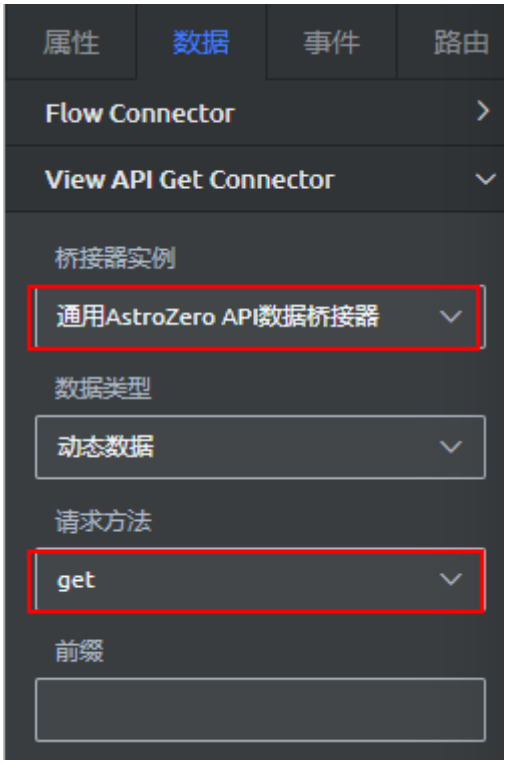
   图 2-39 添加命名空间前缀

   “login API”为登录接口的URL后半段，请根据实际情况修改，调用服务编排时，会在服务编排名称前面自动拼接租户命名空间的前缀，登录接口将会在**创建用户登录脚本**章节创建。

2. 在“数据”页签，单击“View API Get Connector”，设置“桥接器实例”为“通用AstroZero API数据桥接器”，“数据类型”为“动态数据”，“请求方法”为“get”，如**图2-40**所示。

图 2-40 设置桥接器



自定义组件的“数据”参数说明如下：

- 桥接器实例：调用的桥接器名称。
- 请求方法：调用的方法名，如get（查询）、put（增加）、post（修改）和delete（删除）。
- 调用周期：此处不用配置。调用周期是每隔多少秒调用一次后台接口或获取静态数据，默认配置为“0”，表示只调用一次或只获取一次静态数据。

3. 参考上一步，分别设置如图2-41所示的其他桥接器实例。

图 2-41 设置其他桥接器

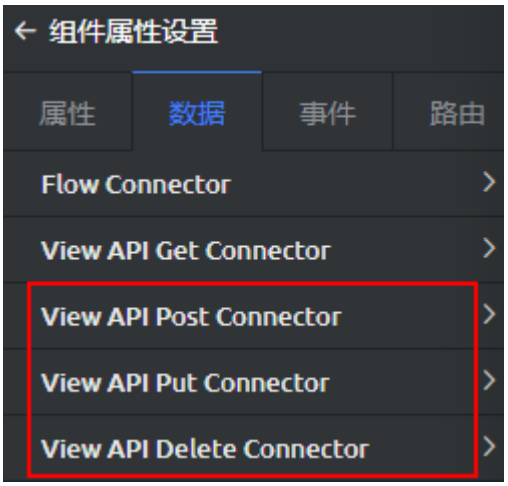


表 2-4 桥接器实例配置

数据名	桥接器实例	数据类型	Request Method
View API Get Connector（上一步已配置）	通用AstroZero API数据桥接器	动态数据	get
View API Post Connector	通用AstroZero API数据桥接器	动态数据	post
View API Put Connector	通用AstroZero API数据桥接器	动态数据	put
View API Delete Connector	通用AstroZero API数据桥接器	动态数据	delete

步骤5 设置自定义组件“userLogin”的“事件”，使自定义组件与其他页面关联。

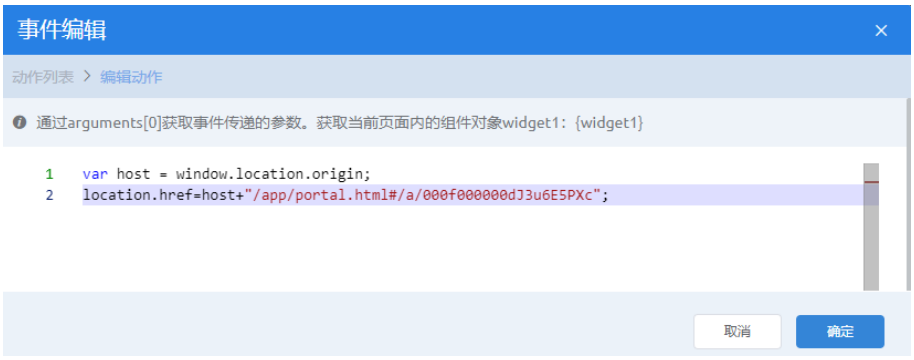
图 2-42 需要配置的事件

1. 单击选中“userLogin”组件，在右侧属性设置的“事件”页签下，单击“go Homepage”后的。
2. 单击“新建动作”，选择“自定义 > 自定义动作”。

图 2-43 配置事件

3. 在“事件编辑”中，输入事件代码，然后单击“确定”。

图 2-44 事件编辑



事件代码如下，其中加粗斜体代码请修改为实际运行环境主页URL：

```
var host = window.location.origin;
location.href=host+"/app/portal.html#/a/000f000000dJ3u6E5PXc";
```

其中，“/app/portal.html#/a/**000f000000dJ3u6E5PXc**”为当前应用运行环境Home页URL，可在预览应用中查看。即在左侧导航栏中，单击  ，进入应用预览，查看地址栏红框中URL。

图 2-45 应用预览入口

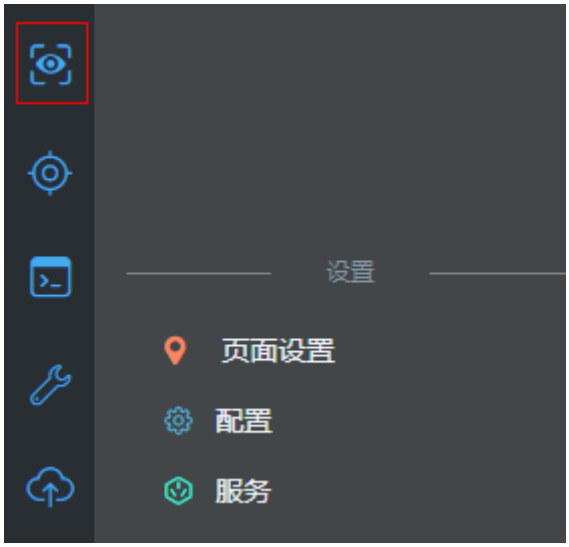


图 2-46 查看运行环境 Home 页 URL

步骤6 单击页面上部的，保存页面。

步骤7 单击页面上部的，发布页面。

因为页面中配置的跳转目标页面、登录业务逻辑、相关账号等，当前还没有创建，因此，预览时，仅能查看登录页面的前端样式，不能进行登录操作。

说明

当前登录页中，输入业务账号及密码，单击“登录”按钮的登录逻辑是通过“自定义登录”组件，调用用户登录业务逻辑完成的。相关的业务逻辑（脚本、服务编排），将在**开发业务逻辑**中创建。

----结束

2.6 设备管理开发

2.6.1 开发前必读

设备管理功能主要是通过操作两个标准页面（编辑设备、设备管理），来实现电梯设备信息的新增、修改、删除、查询及监控等功能的。

设备管理界面的大致构想，如**图2-47**所示。界面上默认显示应用中保存的所有电梯设备信息，可以直接新增、修改或者删除电梯记录，也可以查询部分电梯记录。

- 单击“新增设备”按钮，可以在界面上插入一个空行，输入内容后单击“保存”，即可新插入一条电梯记录。
- 直接编辑表格中任意内容，单击“保存”，即可修改任意一条电梯记录。
- 选中记录，单击“删除”，即可删除任意一条电梯记录。
- 设置查询条件，单击“查询”，可以查询满足条件的电梯记录。

图 2-47 电梯设备信息管理界面

通过开发设备管理功能，带您学习如何在AstroZero中，开发脚本、服务编排、标准界面和高级界面，并在开发过程中，细致解析设备管理的增加、删除和修改，以及设备位置信息监控功能开发过程及原理。

2.6.2 定义数据对象

2.6.2.1 背景和原理（对象）

AstroZero提供的数据对象（Object）定义功能，对应传统方式开发业务系统中的创建数据库表。每个Object对应一张数据库表，用于保存业务系统需要的配置数据和业务数据。

对象用于存储组织或者业务特有的数据，可理解为数据库中的数据表（逻辑表，系统实际存储时通过字段映射，统一把数据保存在数据库大宽表中）。

您可以围绕对象这一核心，定义相关的字段、字段校验规则、界面样式、字段变更时的触发事件等。如果把待开发的业务系统比作一部电影，对象就是电影中的一个个角色，需要勾勒角色的外貌、性格特点、人物关系和所经历的剧情。

租户开发者可以自定义对象（Custom Object），平台允许租户增、删、改自定义对象，以及对象的字段，需要注意的是同一个租户账号下，创建的自定义对象的名称唯一不能重复。同时，平台上也有预置对象（Standard Object），您可以为这些标准对象新增字段，但不能修改、删除基线字段。

当自定义对象时，AstroZero会为这些Object自动创建一些标准字段（Standard Fields），如图2-48所示。

图 2-48 自定义对象的标准字段

您自定义的字段将保存在上图中的“自定义字段”页签，更多关于对象模型的详细介绍请查看**定义对象**。

学习地图

如**图2-49**所示，通过本节的学习和实践，您可以初步了解“对象”和“标准页面”的基本概念和能力。

图 2-49 学习地图

2.6.2.2 方法和实践

对于设备管理功能，需要先创建一个设备对象Equipment，保存设备品牌、型号、资产编号等信息，如**表2-5**所示。

表 2-5 设备对象 Equipment 信息

字段标签	字段名称	字段描述	数据类型	是否唯一	是否必填	建议长度
equipmentName	name（复用平台预置的标准字段）	设备名称	Name	是	NA	NA

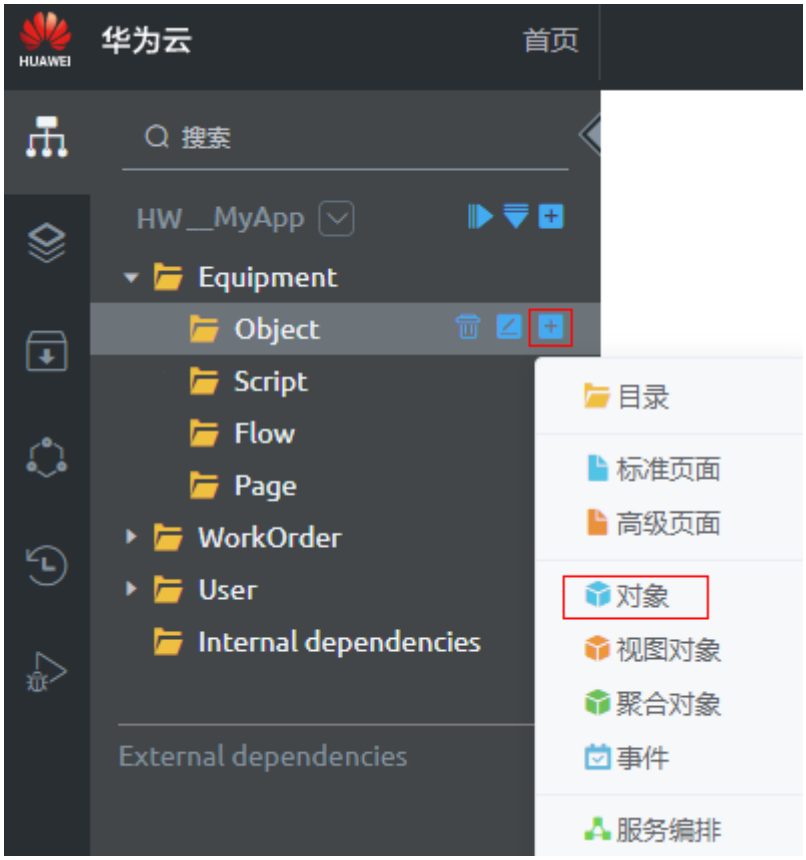
字段标签	字段名称	字段描述	数据类型	是否唯一	是否必填	建议长度
latitude	latitude	设备位置纬度	文本	否	否	255
fullAddress	fullAddress	完整地址	文本	否	否	255
installationSiteCity	installationSiteCity	设备所在市	文本	否	否	255
installationSiteArea	installationSiteArea	设备所在区	文本	否	否	255
installationDetailAddress	installationDetailAddress	设备所在详细地址	文本	否	否	255
longitude	longitude	设备位置经度	文本	否	否	255
equipmentSN	equipmentSN	设备编码	文本，区分大小写	是	是	255
equipmentBrand	equipmentBrand	设备品牌	选项列表 预置选项包括沃克斯、三菱、日立、迅达等。	否	否	-
equipmentModel	equipmentModel	设备型号	文本	否	否	255
installationSiteProvince	installationSiteProvince	设备所在省	文本	否	否	255

操作步骤

步骤1 在AstroZero经典版开发环境首页“项目 > 我的应用”中，单击“设备维修管理系统”，进入之前创建的应用。

步骤2 在“Equipment”中，将鼠标放在“Object”目录上，单击界面上出现的“+”，在弹出菜单中选择“对象”。

图 2-50 创建对象



步骤3 选中“创建新对象”，在“标签”和“名称”文本框中输入“Equipment”，单击“添加”。

系统实际创建的对象名称为“***HW***__Equipment__CST”，“***HW***__”前缀由租户命名空间namespace决定，“__CST”后缀代表自定义对象。在AstroZero中，同一个租户账号下，创建的自定义对象“名称”不能重复。

步骤4 将标准字段“Name”作为“设备名称”字段。

1. 在“标准字段”页签，单击“Name”字段后的编辑图标。
2. 如图2-51所示，修改字段标签为“equipmentName”，并设置字段内容唯一，单击“保存”。

图 2-51 修改 name 字段的标签

步骤5 按系统向导完成第1个字段“设备编码”（equipmentSN）的定义。

1. 在“自定义字段”页签，单击“新建”。

图 2-52 新建自定义字段

2. 在选中字段类型中，设置字段类型为“文本”，单击“下一步”。
3. 输入字段详细信息，如图2-53所示，设置字段标签、名称为“equipmentSN”、字段长度255，并定义字段为必填字段、是唯一且大小写敏感，单击“下一步”。

注意

平台会根据字段标签自动生成字段名称，但请参照表2-5，修改字段名。对于utf-8编码，一个汉字占用三个字节。

图 2-53 定义“设备编码”详细信息

系统实际创建的字段名称为“***HW***__equipmentSN__CST”，“***HW***__”前缀由租户命名空间namespace决定，“__CST”后缀代表自定义字段。

4. 配置字段级安全。选中“读取”和“编辑”复选框，为所有预置profile配置能编辑和读取本字段的权限，单击“下一步”。
5. 将字段添加到页面布局。选中“添加本字段到该页面布局”，单击“保存”。

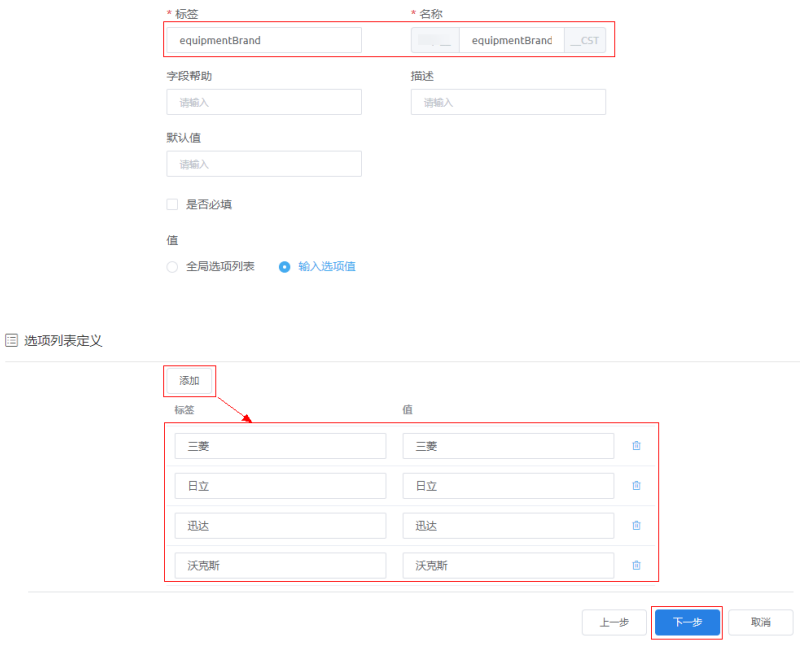
步骤6 您可按系统向导完成第2个字段“设备品牌”（equipmentBrand）的定义。

1. 在“自定义字段”页签，单击“新建”。
2. 设置字段类型为“选项列表”，单击“下一步”。
3. 如图2-54所示，设置字段标签、名称，输入可选值，添加选项列表的标签和值，单击“下一步”。

说明

AstroZero会根据字段标签自动生成字段名称，但请参照表2-5，修改字段名。

图 2-54 定义“设备品牌”详细信息



4. 选中“权限集”后“读取”和“编辑”复选框，为所有预置profile配置能编辑和读取本字段的权限，单击“下一步”。

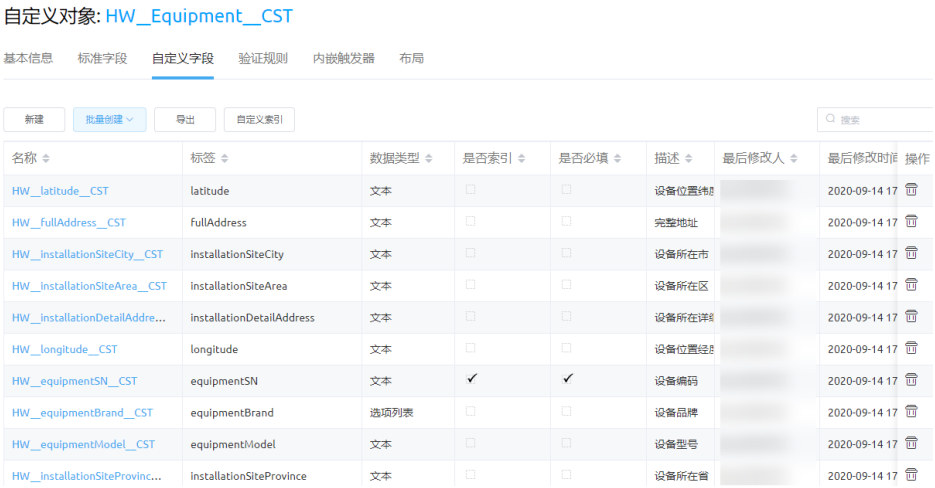
5. 选中“添加本字段到该页面布局”，单击“保存”。

步骤7 创建表2-5中剩余字段。这里有两种方法，请选择合适的方式进行创建。

- 方法一：参考上一步，分别创建表2-5中其他属性。
- 方法二：通过导入字段模板，批量快速创建剩余字段模板。
  a. 单击import-fields-equipment.xlsm，将导入字段模板下载到本地。
  b. 单击当前对象的“自定义字段”，然后单击“批量创建”后的，选择“批量导入字段”，选择步骤7.a中下载的字段模板。导入完成后，刷新页面，重新进入对象的“自定义字段”，即可查看已导入的字段。

图 2-55 导入自定义字段

图 2-56 查看已创建自定义字段



----结束

验证

您可以用对象的页面布局Layout能力，检查对象的定义是否符合预期。

创建对象时，系统会自动为对象创建两个页面布局：

- 设备详情页面Equipment Detail：显示对象中单条记录的详细信息。如果手工创建字段时选中了“添加本字段到该页面布局”，设备详情页面包含name和添加的自定义字段。脚本创建的默认不添加，但是不影响后续的操作。

- 设备列表页面Equipment Records：显示对象中的记录列表。列表页面中默认只显示了name字段（设备名称）。因为后续章节的测试中，经常需要检查记录ID、设备编码和设备名称，所以将其他几个字段也添加到列表页面中。

步骤1 修改设备对象的列表页面布局。

1. 单击设备对象HW__Equipment__CST的“布局”页签。
2. 单击“Equipment Records”，打开页面详情。

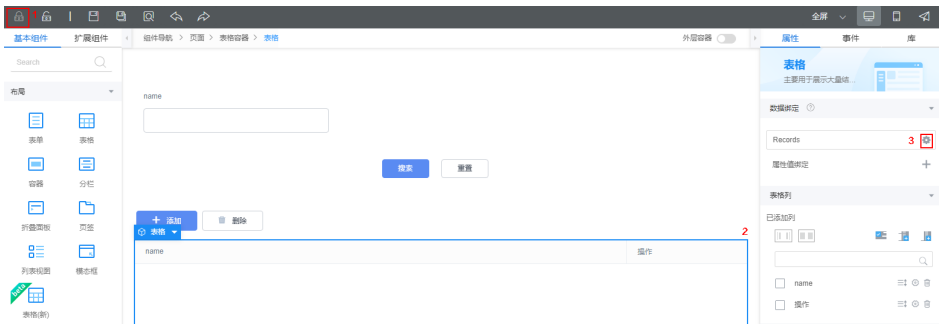
图 2-57 对象布局

3. 选中列表“表格”，如图2-58所示，再单击界面右侧“属性”页签中的。

说明

如果当前页面已锁定，请先单击，解锁页面。

图 2-58 绑定 Records

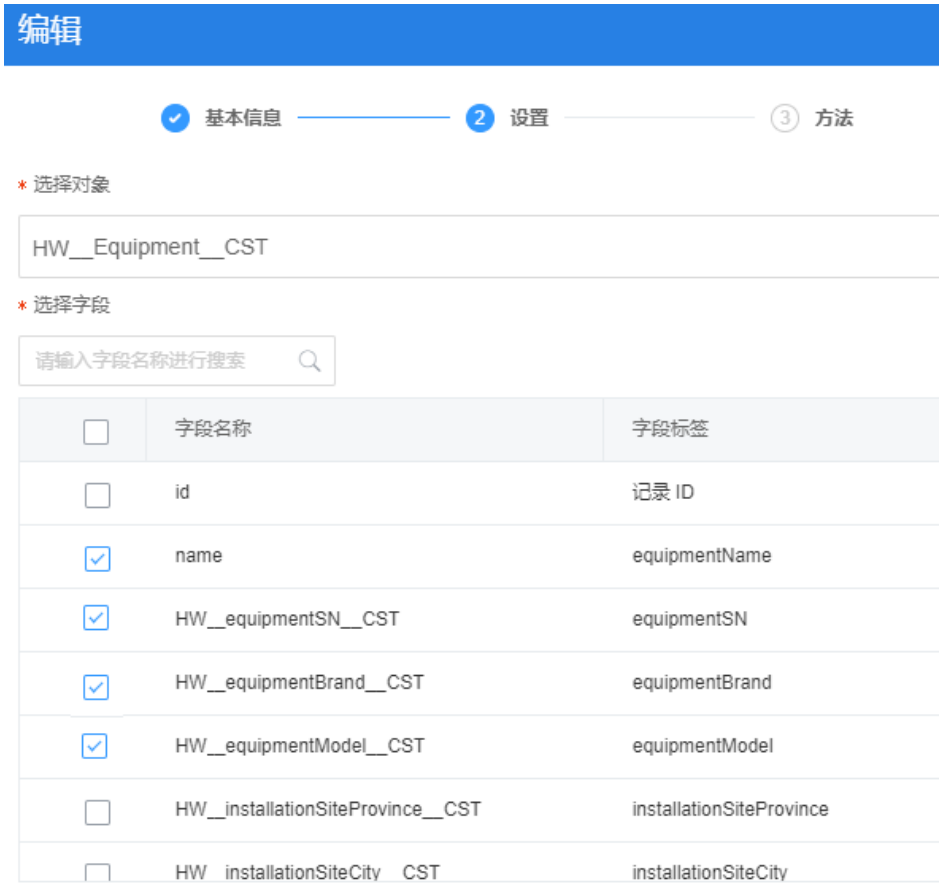


4. 单击Records后的编辑图标。

图 2-59 编辑 Records

5. 选择对象及要显示在页面上的字段，如图2-60所示，单击“下一步”，再单击“确定”。

图 2-60 为 Records 添加字段



6. 在图2-59对话框中，单击选中“Records”，再单击“确定”。
7. 单击编辑器上方的，保存设置。

步骤2 预览、测试对象。

1. 单击页面上的，进入预览页面。
2. 单击“添加”，再输入任意测试数据，单击“保存”。

系统返回页面，并显示新插入的数据，则表示对象创建成功。

----结束

2.6.3 开发“编辑设备”功能

2.6.3.1 背景和原理

本节主要通过创建一个标准页面，调用一个具有编辑设备功能的脚本，实现编辑设备信息功能。在进行开发前，您需要先了解脚本、公共接口以及标准页面的相关知识。

学习地图

如图2-61所示，通过本章的学习和实践，您将了解“标准页面”的能力，并掌握脚本的开发方法。

图 2-61 学习地图

2.6.3.1.1 脚本

对应相对复杂的业务逻辑，AstroZero提供了脚本、服务编排等后台逻辑形式。本章节主要带您了解脚本的基础知识。

AstroZero的脚本引擎采用TypeScript语言。脚本执行时，TypeScript语言会被翻译成JavaScript语言，由JavaScript引擎执行。在JavaScript es5的官方标准库外，AstroZero还扩展了10+内置功能库，帮助您更高效地开发，如表2-6所示。除此之外，AstroZero还提供了代码智能补全、代码格式化、语法错误提示等功能，帮助您高效编辑代码。

脚本编辑界面如图2-62所示，您可在线编辑、测试和发布脚本。

图 2-62 脚本编辑界面

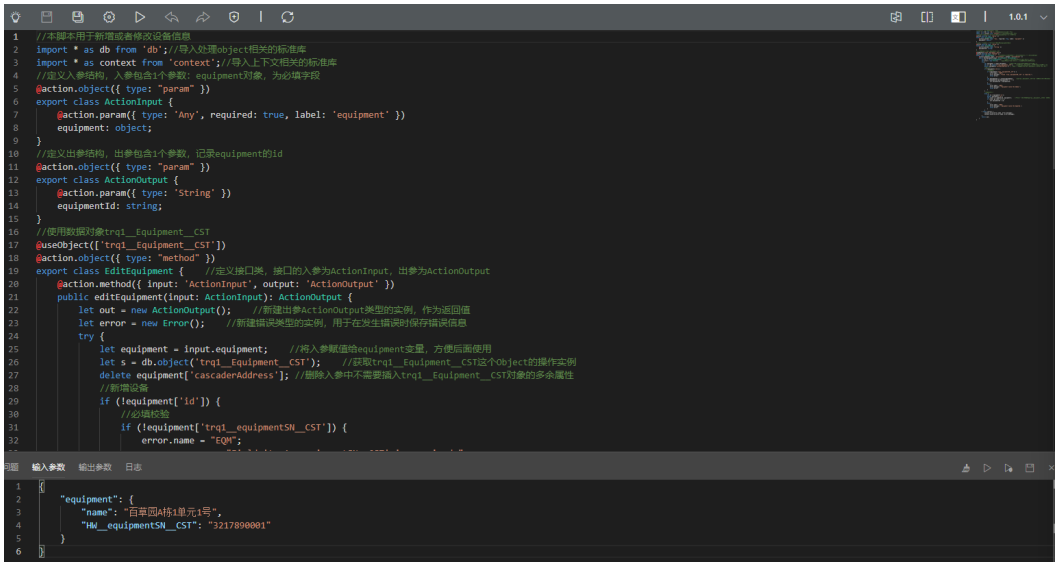


表 2-6 标准库说明

标准库名称	说明
sys	提供系统层面的操作，例如获取系统参数值。
context	脚本执行的上下文，例如获取当前租户ID、获取当前登录用户ID等。
action	定义脚本的输入输出方法，以及调用脚本的方法。
buffer	操作二进制缓存区，例如将缓冲区内容转换为字符串。
setup	对系统对象Standard Object的增、删、改、查。
db	对自定义对象Customer Object的增、删、改、查。
sql	执行sql语句，目前只支持执行select查询语句。
bo	对Business Object的增、删、改、查。
meta	操作object的元数据，例如获取字段名称、获取字段长度。
http	http或者https操作。
iconv	将字符串转为GBK、GB18030、GB2132或ISO8859_1编码格式，或者解码字符串。
crypto	加密操作，包含对哈希、HMAC、加密、解密、签名及验证的封装。
decimal	将从http请求或从数据库中查询出来的number类型值，转换成Decimal类型，并能进行加、减、乘、除等数学运算。
excel	操作Excel文件，例如生成Excel文件。
text	操作文本，例如将文本转换为二进制。

标准库名称	说明
xml	操作XML文件，例如读取XML文件内容。
uuid	生成全局唯一标识。

样例代码解读

通过下面详细的脚本代码内容解读，使您对脚本有一个更具体的认识。

一般情况下，编写脚本的大致流程为：

1. 按需引入平台标准库。

   图 2-63 引入平台标准库

```
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import * as decimal from 'decimal';//导入decimal数据类型相关的标准库
```

2. 定义出参、入参结构。

   图 2-64 定义入参

```
//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true })
    equipmentId: string;//设备ID
}
```

   图 2-65 定义出参

```
//定义出参结构
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any', label: 'equipment' })
    equipment: object;//设备对象
}
```

3. 定义方法以及使用的对象。

   图 2-66 定义方法及使用对象

```
@useObject(['HWT__Equipment__CST'])//使用数据库对象Equipment__CST
@action.object({ type: "method" })
export class QueryEquipmentDetail {
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public queryEquipmentDetail(input: ActionInput): ActionOutput {
```

4. 进行数据库操作。

图 2-67 数据库相关操作

```
//获取HWT__Equipment__CST这个Object的操作实例
let s = db.object('HWT__Equipment__CST');
//查询字段(全部)
let option = {};
//查询条件
let condition = {
    "conjunction": "AND",
    "conditions": [{
        "field": "id",
        "operator": "eq",
        "value": input.equipmentId
    }]
};
//调用按条件查询Equipment__CST的接口
let record = s.queryByCondition(condition, option);
```

下面通过解读以下脚本样例，了解一个脚本的总体结构框架、编写要求。

```
import * as decimal from 'decimal';

@action.object({type: "param"})
export class ActionInput {
    @action.param({type: 'String', required: true, label: 'your name', description: 'please input your name'})
    name: string;

    @action.param({type: 'Number', required: true, min: 1, max: 100, message: 'age must during [1, 100]'})
    age: decimal.Decimal;

    @action.param({type: 'Date', pattern: 'yyyy-MM-dd'})
    birthday: Date;

    @action.param({type: 'String', isCollection: true})
    schools: string[];

    @action.param({type: 'Boolean'})
    married: boolean;

    @action.param({type: 'MyObject'})
    obj: MyObject;
}

@action.object({type: "param"})
export class MyObject {
    @action.param({type: 'String'})
    something: string;
     @action.param({type: 'Number'})
    otherthing: decimal.Decimal;
}

@action.object({type: "param"})
export class ActionOutput {
    @action.param({type: 'String', isCollection: true})
    greets: string[];
}

@action.object({type: "method"})
export class ActionDemo {
    @action.method({ label: 'greeting something', description: 'greeting something.', input: 'ActionInput',
output: 'ActionOutput' })
    public greet(inarg: ActionInput): ActionOutput {

        console.log('name = ', inarg.name);
        console.log('age = ', inarg.age);
        console.log('birthday = ', inarg.birthday);
        console.log('schools = ', inarg.schools);
```

```
        console.log('married = ', inarg.married);
        console.log('obj = ', inarg.obj);

        let out = new ActionOutput();
        out.greets = ['hello', 'hi', 'how are you', 'how old are you', 'long time no see'];
        return out;
    }
}
```

上述示例脚本包括三部分内容：

1. 导入标准库或其他模块。

   示例中第1行，表示将使用平台提供的decimal库。

   ```
   import * as decimal from 'decimal';
   ```

   除了平台预置标准库，还可以声明对其他自定义模块的引用。例如，已经提前开发了一个脚本circle，可以用如下方式加载。

   ```
   import * as circle from './circle';
   ```

2. 定义输入、输出变量。

   脚本可以有多个输入、输出参数，也可以没有。所有的输入或输出参数必须封装在一个class中，作为实例成员。

   本例中，脚本有6个输入参数，被封装为ActionInput。每个参数都必须定义其参数类型，同时还可以定义是否必填、标签、最大值、最小值等可选属性。

   ```
   @action.object({type: "param"})
   export class ActionInput {
       @action.param({type: 'String', required: true, label: 'your name', description: 'please input your
   name'})
       name: string;

       @action.param({type: 'Number', required: true, min: 1, max: 100, message: 'age must during [1,
   100]'})
       age: decimal.Decimal;

       @action.param({type: 'Date', pattern: 'yyyy-MM-dd'})
       birthday: Date;

       @action.param({type: 'String', isCollection: true})
       schools: string[];

       @action.param({type: 'Boolean'})
       married: boolean;

       @action.param({type: 'MyObject'})
       obj: MyObject;
   }
   ```

   因为第6个输入参数“obj”的参数类型为自定义对象，所以还需要给出“MyObject”的定义。

   ```
   @action.object({type: "param"})
   export class MyObject {
       @action.param({type: 'String'})
       something: string;
        @action.param({type: 'Number'})
       otherthing: decimal.Decimal;
   }
   ```

   脚本有1个输出参数，被封装为ActionOutput。

   ```
   @action.object({type: "param"})
   export class ActionOutput {
       @action.param({type: 'String', isCollection: true})
       greets: string[];
   }
   ```

3. 定义方法。

样例中，ActionDemo是外部调用的class，使用export导出。ActionDemo定义了一个action method，使用action.method装饰，表明调用脚本时从此方法入口。greet是class的实例方法，其输入、输出参数就是前面定义的ActionInput和ActionOutput。

在一个脚本文件里面，action.method只能使用一次。

```
@action.object({type: "method"})
export class ActionDemo {
    @action.method({ label: 'greeting something', description: 'greeting something.', input: 'ActionInput', output: 'ActionOutput' })
    public greet(inarg: ActionInput): ActionOutput {

        console.log('name = ', inarg.name);
        console.log('age = ', inarg.age);
        console.log('birthday = ', inarg.birthday);
        console.log('schools = ', inarg.schools);
        console.log('married = ', inarg.married);
        console.log('obj = ', inarg.obj);

        let out = new ActionOutput();
        out.greets = ['hello', 'hi', 'how are you', 'how old are you', 'long time no see'];
        return out;
    }
}
```

脚本编辑页面不支持单步调试，样例里的console.log可实现在日志里打印过程输出，方便代码调试。

2.6.3.1.2 公共接口

公共接口是对脚本、服务编排和对象进行再包装的一种方式。将创建的脚本、服务编排、对象包装成一个新公共服务，可以使得接口的URL地址的表达形式更规范，方便让前端页面或第三方系统进行调用。

因此，在创建完后台逻辑后（服务编排、脚本、对象），就需要先将此接口包装成标准的公共接口，才能被调用。本章节中创建了2个业务脚本，因此需要对应创建2个公共接口，供前端页面调用。

创建公共接口的入口：在APP视图下，单击下方“服务”，即可进入公共接口创建页面。

图 2-68 创建公共接口入口

图 2-69 公共接口基本信息

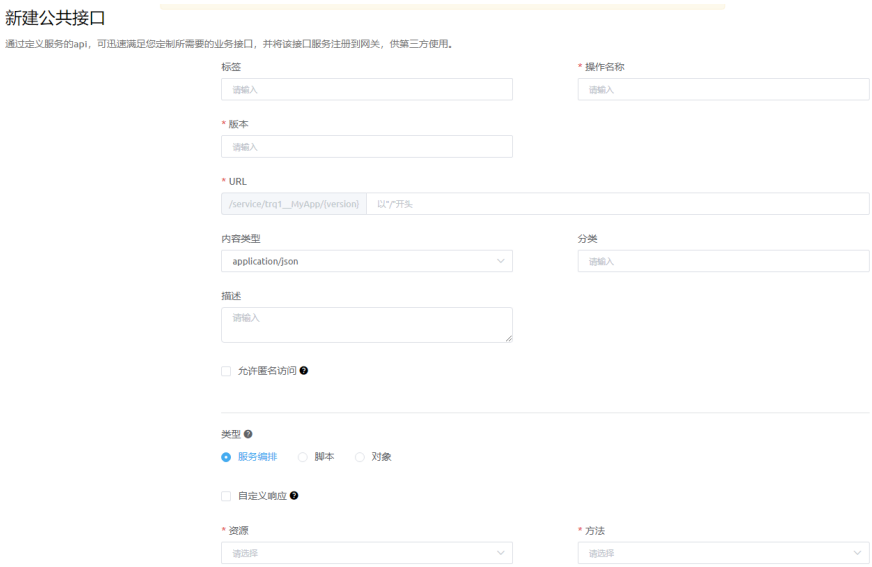


图 2-70 公共接口 URL

2.6.3.1.3 标准页面

相对于高级页面，标准页面主要用于对前端页面的样式要求相对简单的场景，这种页面一般只是针对业务数据的增、删、改、查等基础功能。

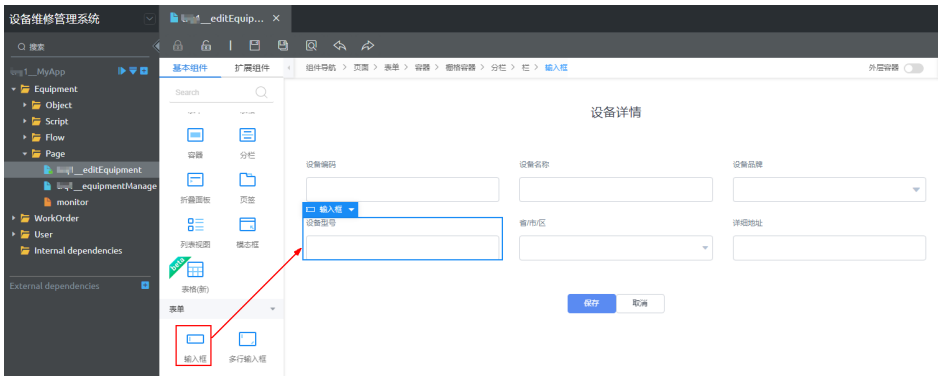
标准页面主要是通过对前端页面组件的组合编排，以及为组件配置属性、关联事件实现业务功能的，在**组装“编辑设备”页面**章节，则会详细介绍组件、组件属性以及事件之间的调用及相关配置。

标准页面的编辑环境也称为UI Builder，UI Builder是图形化、无码化的前端页面在线开发工具。借助标准页面，您只需要了解基本前端编码知识，通过将页面组件拖拽至“设计视图”的画布中，再进行必要的属性、事件配置，就可以完成页面的开发。

设计视图

标准页面中预置了多种组件，在开发页面时，可以直接从左侧组件区域，将这些组件拖拽到右侧“页面内容”中。

图 2-71 标准页面编辑界面（UI Builder）



配置组件属性

- 了解组件使用场景及配置方法

  在配置使用组件前，可以将光标放在每个组件上后，组件右上角将显示帮助信息的问号图标。单击问号图标，即可进入该组件介绍页面，了解并学习AstroZero预置前端组件的使用场景及参数配置方法。

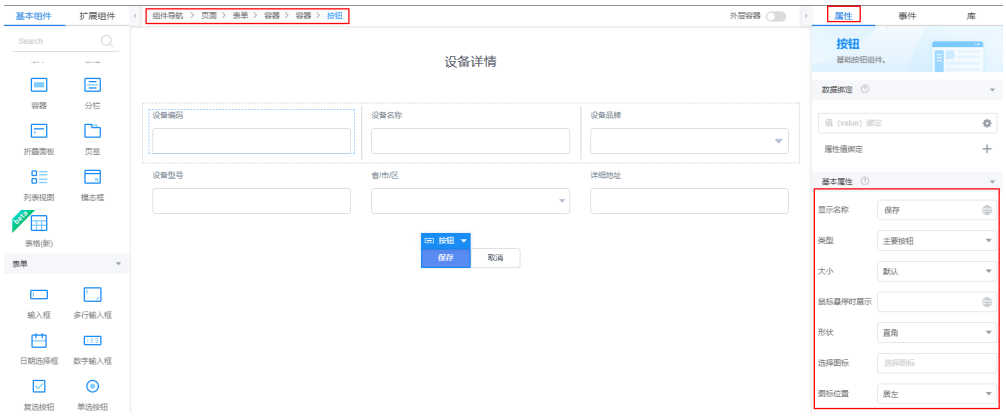
  图 2-72 组件帮助信息入口

- 选中组件，查看组件属性

  在“设计视图”中，选中一个页面组件，可以在右侧“属性”页签，设置该组件的绑定数据、样式等属性，如图2-73所示。在设计视图中选中一个文本框，可以在右侧修改这个文本框的标签名、样式、绑定的数据模型等。

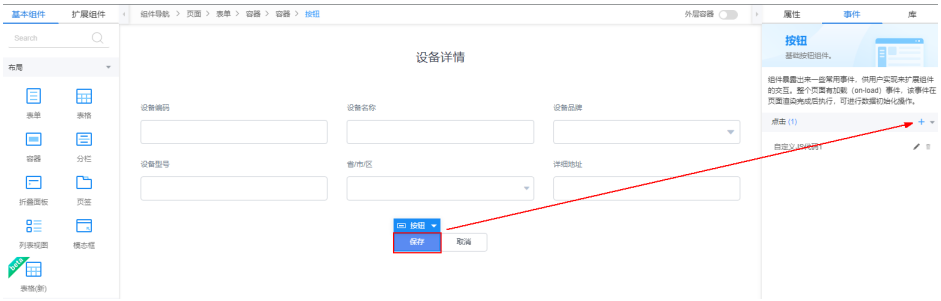
  图 2-73 设置组件的属性

- 页面及组件的事件代码

  在“设计视图”中，选中任意组件，可以在右侧“事件”页签，设置组件的关联事件。例如，可以选择一个“按钮”，然后在按钮的“点击”事件中，添加相应的事件代码，实现在单击按钮后将界面输入保存到数据库中。

图 2-74 编辑组件的关联事件



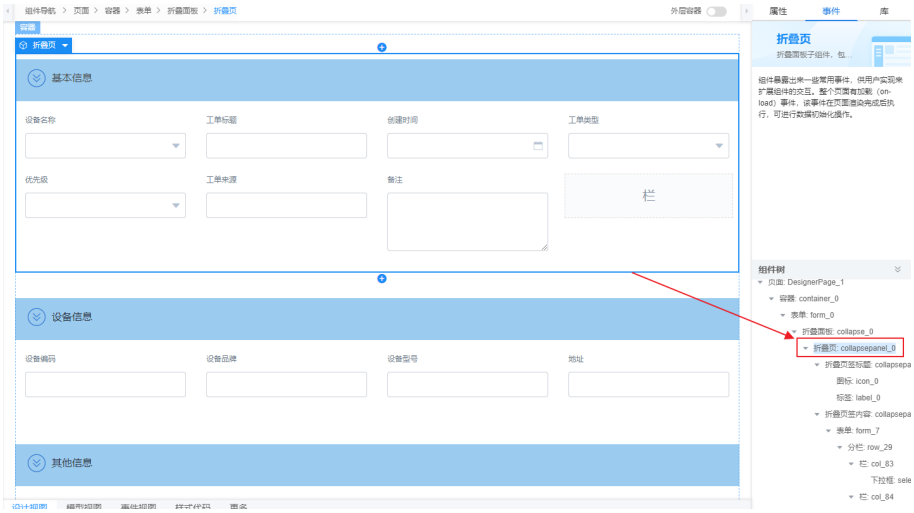
- 利用组件导航，快速选中组件

  当选中某个组件时，页面上方会在组件导航上显示它的html标签层级。因此，在组件数量比较多，位置较为紧密时，可以直接单击标签层级，快速选择组件，也可以快速切换组件。例如，在上图的组件导航中单击“页面”，可以直接选中页面上的最外层页面组件。

- 利用组件树，快速选中组件

  在组件数量比较多，位置较为紧密时，也可以单击页面右下角的“组件树”，展开组件树，在组件树中，直接单击标签层级，快速选择组件，也可以快速切换组件。

图 2-75 利用组件树快速选中组件

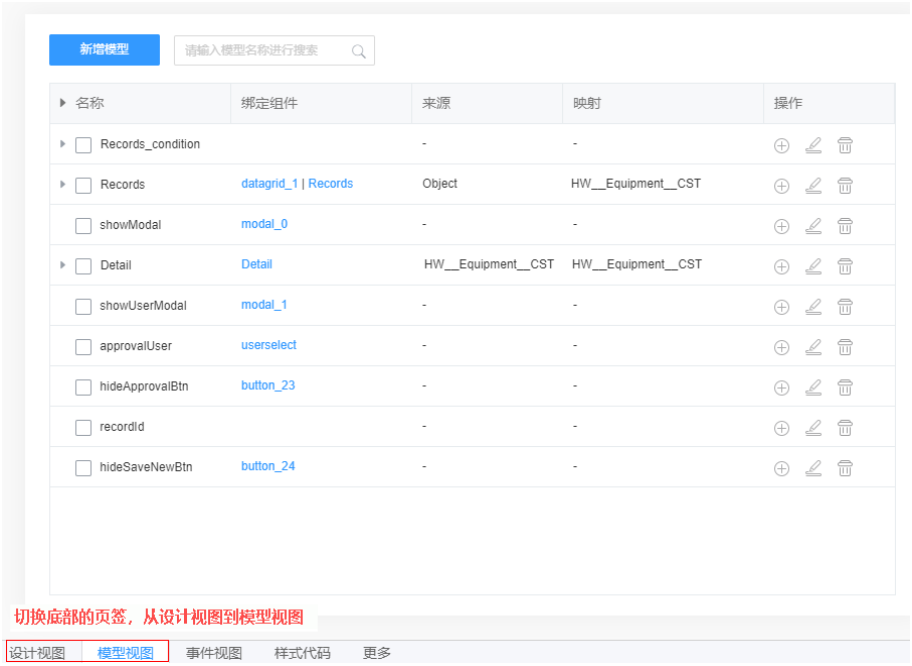


模型视图

标准页面是通过数据模型驱动的，页面所有的逻辑都是围绕数据模型展开的。在将模型与前台页面组件或者后台逻辑绑定后，开发者只需要关注模型数据的实例化和处理，不需要关注页面的渲染和展示。

在页面底部，单击“模型视图”，即可从“设计视图”切换到“模型视图”，在模型视图下，可以查看、编辑以及管理数据模型，如图2-76所示。

图 2-76 模型视图



模型定义

标准页面支持表2-7中四类模型，每类模型都包含参数定义和方法定义。方法是在模型上定义的API，通常会在前端组件关联的事件脚本（例如页面加载事件、鼠标单击事件）中调用这些API，以实现一定的逻辑。

表 2-7 模型说明

模型分类	模型说明	模型参数的定义	模型方法的定义	API调用方法
自定义模型	自定义模型是由开发者自由定义的模型。	参数由开发者自定义，可以添加子节点。	开发者自定义方法。	*$model.ref("modelName").actionName();*
对象模型	对象模型是直接与Object对象表映射生成的。	系统自动获取Object所有的字段，开发者可以从中选择部分字段作为参数。	平台自动生成查询、保存、删除和统计4个方法。	• *$model.ref("modelName").query(param);* • *$model.ref("modelName").save();* • *$model.ref("modelName").delete();* • *$model.ref("modelName").count();*

模型分类	模型说明	模型参数的定义	模型方法的定义	API调用方法
服务模型	服务模型是与后台服务映射生成的，当前支持与服务编排或Script映射。	参数根据后台服务的入参、出参映射生成为inputParam和outputParam节点。	平台自动生成了run方法，用于执行模型关联的服务编排或者Script。	*$model.ref("modelName").run();*
事件模型	事件模型是与后台事件的字段映射生成的，并且支持websocket刷新模型数据。	参数根据后台事件的字段映射生成。	平台自动生成了run方法，用于执行模型关联的服务编排或者Script。	*$model.ref("modelName").run();*

除了在模型中定义的方法，AstroZero还提供了如下标准API：

- 获取模型数据：*$model.ref("modelName").getData();*
- 设置模型数据：*$model.ref("modelName").setData();*
- 设置模型字段值：*$model.ref("modelName").setValue(key,value);*

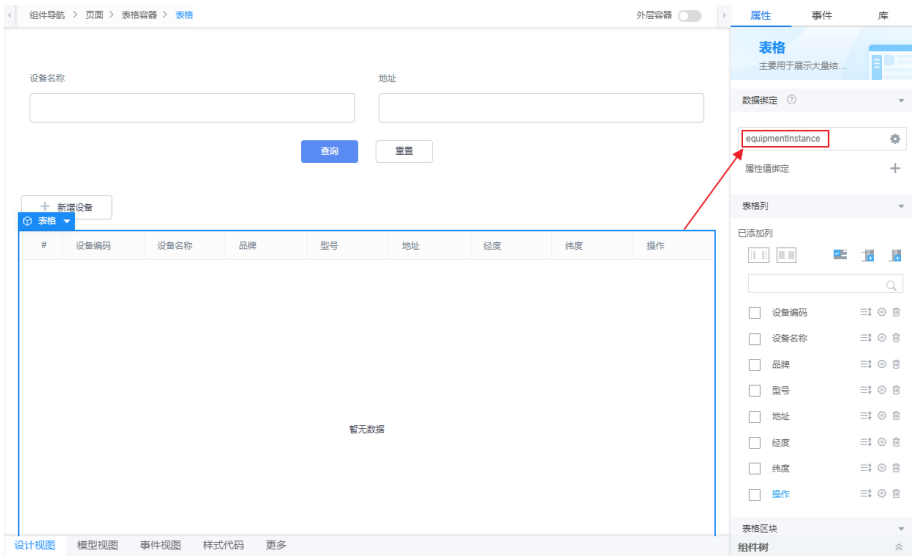
模型绑定

模型与前端组件的绑定分为值绑定和属性绑定，绑定会在指定组件上创建双向数据绑定。

- 典型的值绑定场景有：表单、表格、列表视图对应的model绑定，以及输入框、下拉框等基础组件对应的value绑定，类似Vue的v-model。
- 典型的属性绑定场景有：下拉框的选项值、步骤条的步骤值等，类似Vue的v-bind。

如图2-77所示的例子中，实现了表格DataGrid组件与值的绑定。

图 2-77 数据绑定



2.6.3.2 创建业务逻辑

“编辑设备”功能主要通过调用并执行一个具有“编辑设备”功能的脚本，实现编辑设备信息功能，并在修改设备信息时，能够根据设备id号，直接修改设备信息，因此需要在开发页面前，先创建一个“编辑设备”和“按ID查询设备详情”脚本。

2.6.3.2.1 创建“编辑设备”脚本

“编辑设备”脚本实现的功能是向设备对象中插入1条新记录，或者更新1条记录。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“editEquipment”，单击“添加”。

系统实际创建的脚本名称为“***HW***__editEquipment”，“***HW***__”前缀由租户命名空间namespace决定。新建创建的脚本，默认是当前用户锁定状态，可以进行编辑保存等操作。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击，进行锁定。

步骤4 在代码编辑器中，插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于新增或者修改设备信息
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
```

```
//定义入参结构，入参包含1个参数：equipment对象，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'Any', required: true, label: 'equipment' })
    equipment: object;
}
//定义出参结构，出参包含1个参数，记录equipment的id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    equipmentId: string;
}
//使用数据对象HW__Equipment__CST
@useObject(['HW__Equipment__CST'])
@action.object({ type: "method" })
export class EditEquipment {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public editEquipment(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let equipment = input.equipment;    //将入参赋值给equipment变量，方便后面使用
            let s = db.object('HW__Equipment__CST');    //获取HW__Equipment__CST这个Object的操作实例
            delete equipment['cascaderAddress']; //删除入参中不需要插入HW__Equipment__CST对象的多余属性
            //新增设备
            if (!equipment['id']) {
                //必填校验
                if (!equipment['HW__equipmentSN__CST']) {
                    error.name = "EQM";
                    error.message = "Field 'HW__equipmentSN__CST' is required.";
                    throw error;
                }
                let equipmentId = s.insert(equipment);    //向HW__Equipment__CST插入一条数据，返回数据的唯
一标识即设备ID
                if (equipmentId && equipmentId != "") {
                    out.equipmentId = equipmentId;
                }
                else {
                    error.name = "EQM";
                    error.message = "Equipment Cannot Be Added.";
                    throw error;
                }
            }
            //编辑修改设备
            else {
                let id = equipment['id'];
                delete equipment['id'];
                let count = s.update(id, equipment);    //根据设备ID，编辑更新HW__Equipment__CST的一条数据
                if (count && count == 1) {
                    out.equipmentId = id;
                }
                else {
                    error.name = "EQM";
                    error.message = "Equipment Cannot Be Updated.";
                    throw error;
                }
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方的  ，保存脚本。

----结束

验证并发布

步骤1 测试新增逻辑能否正常执行。

1. 单击编辑器上方的，执行脚本。
2. 如图2-78所示，在界面底部输入测试数据，单击测试窗口右上角执行图标。

图 2-78 测试脚本

测试报文采用json格式，样例如下（报文中加粗斜体内容请替换为实际的字段名）：

```
{
   "equipment": {
      "name": "百草园A栋1单元1号",
      "HW__equipmentSN__CST": "3217890001"
   }
}
```

执行成功，会在“输出”页签返回equipmentId。请保存这个返回结果，后续的测试中会用到。

```
{
   "equipmentId": "cQue000000e1qnhgtCng"
}
```

如果执行失败，请检查之前设备对象、脚本，以及测试报文三者中的对象名、字段名是否一致。

步骤2 到设备对象布局页面（Equipment Records），预览页面，检查数据是否插入成功。

1. 在“Equipment”目录的“Object”下，单击设备对象“***HW***__Equipment__CST”，在“布局”页签下，单击“Equipment Records”后的预览图标。

图 2-79 对象布局页面

2. 在页面中，检查设备列表中是否包含刚插入的测试数据。
3. 如果已新增数据，为了后续测试方便，建议多创建几条数据。

步骤3 测试修改逻辑能否正常执行。

1. 单击编辑器上方的，执行脚本。

2. 在界面底部输入测试数据，单击测试窗口右上角  执行图标。

须知

如下样例报文中的加粗id值，请修改为图2-78的返回结果，加粗斜体字段名请替换为实际的字段名。以下报文是修改name字段。

```
{
   "equipment": {
      "id": "cQuXXXXXXXXng",
      "name": "百草园B栋2单元2号",
      "HW__equipmentSN__CST": "3217890001"
   }
}
```

3. 刷新设备对象布局页面（Equipment Records）的预览页面，查看测试数据是否符合预期。

步骤4 测试成功，单击编辑器上方的  ，启用脚本。

----结束

2.6.3.2.2 创建“按 ID 查询设备详情”脚本

修改已存在的设备信息，则需要创建可以根据设备ID查询出设备详情的脚本。

操作步骤

步骤1 进入创建“设备维修管理系统”应用中创建的应用。

步骤2 在“Equipment”中，将鼠标放在“Script”目录上，单击界面上出现的“+”，选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“queryEquipmentDetail”，单击“添加”。

系统实际创建的脚本名称为“***HW***__queryEquipmentDetail”，“***HW***__”前缀由租户命名空间namespace决定。新建创建的脚本，默认是当前用户锁定状态，可以进行编辑保存等操作。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
/*****************************
 * 本脚本用于按设备ID查询设备详情
 * ***************************/
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
```

```
    @action.param({ type: 'String', required: true })
    equipmentId: string;//设备ID
}
//定义出参结构
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any', label: 'equipment' })
    equipment: object;//设备对象
}
@useObject(['HW__Equipment__CST'])//使用数据库对象HW__Equipment__CST
@action.object({ type: "method" })
export class QueryEquipmentDetail {
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public queryEquipmentDetail(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            //必填校验
            if (!input.equipmentId || input.equipmentId == "") {
                error.name = "EQM";
                error.message = "Equipment id is required.";
                throw error;
            }
            //获取HW__Equipment__CST这个Object的操作实例
            let s = db.object('HW__Equipment__CST');
            //查询字段(全部)
            let option = {};
            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "id",
                    "operator": "eq",
                    "value": input.equipmentId
                }]
            };
            //调用按条件查询Equipment__CST的接口
            let record = s.queryByCondition(condition, option);
            //如果查询到数据
            if (record && record[0]) {
                //拼接前台省市区级联选择器的数据模型
                let cascaderAddress = [];
                if (record[0].HW__installationSiteProvince__CST) {
                    cascaderAddress.push(record[0].HW__installationSiteProvince__CST);
                    if (record[0].HW__installationSiteCity__CST) {
                        cascaderAddress.push(record[0].HW__installationSiteCity__CST);
                        if (record[0].HW__installationSiteArea__CST) {
                            cascaderAddress.push(record[0].HW__installationSiteArea__CST);
                        }
                    }
                }
                record[0].cascaderAddress = cascaderAddress;
                //将结果挂入输出对象中
                out.equipment = record[0]
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方的，保存脚本。

----结束

验证并发布

步骤1 测试新增逻辑能否正常执行。

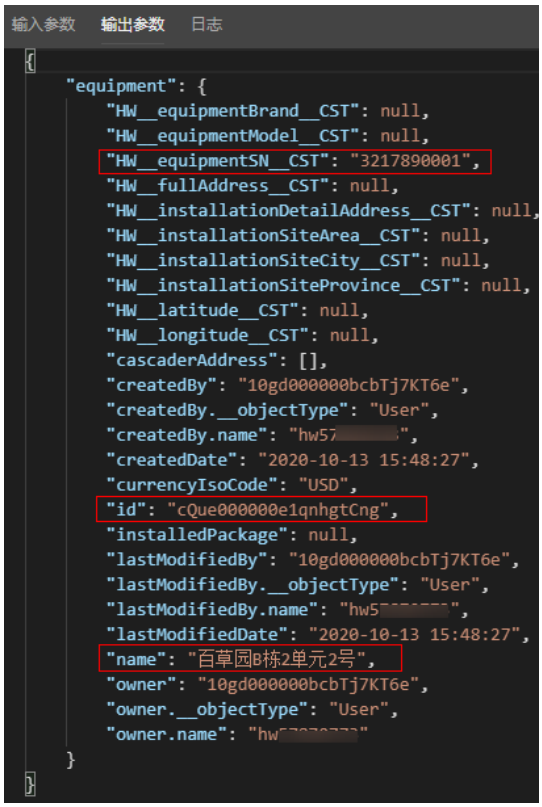
1. 单击编辑器上方的，执行脚本。
2. 在界面底部输入测试数据，单击测试窗口右上角执行图标。

   测试报文样例如下，equipmentId可参考**创建“编辑设备”脚本**验证新增设备时生成的设备ID：

   ```
   {
     "equipmentId": "cQue000000e1qnhgtCng"
   }
   ```

   执行成功，会在“输出”页签返回查询结果，返回结果是上一节中新增的设备信息。

   图 2-80 查询返回的设备信息



步骤2 测试成功，单击编辑器上方的，启用脚本。

----结束

2.6.3.2.3 创建公共服务接口

前端页面支持通过页面创建的服务对象调用脚本，但是为了避免各种权限之间的配置，这里通过创建与脚本“***HW***__editEquipment”、“***HW***__queryEquipmentDetail”一一对应的公共接口，让页面直接调用这种公共接口。

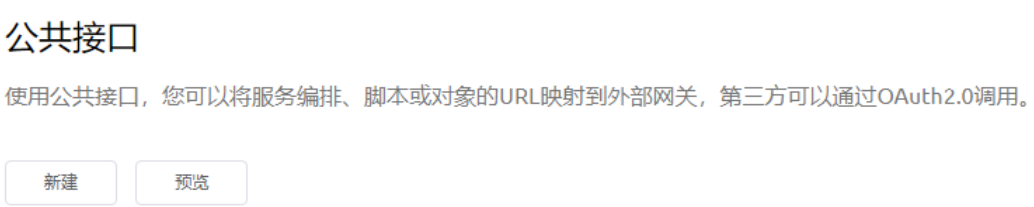
操作步骤

步骤1 在应用中，单击下方“服务”，进入公共接口创建页面。

图 2-81 创建公共接口入口

步骤2 单击“新建”，进入新建公共接口页面。

图 2-82 公共接口创建



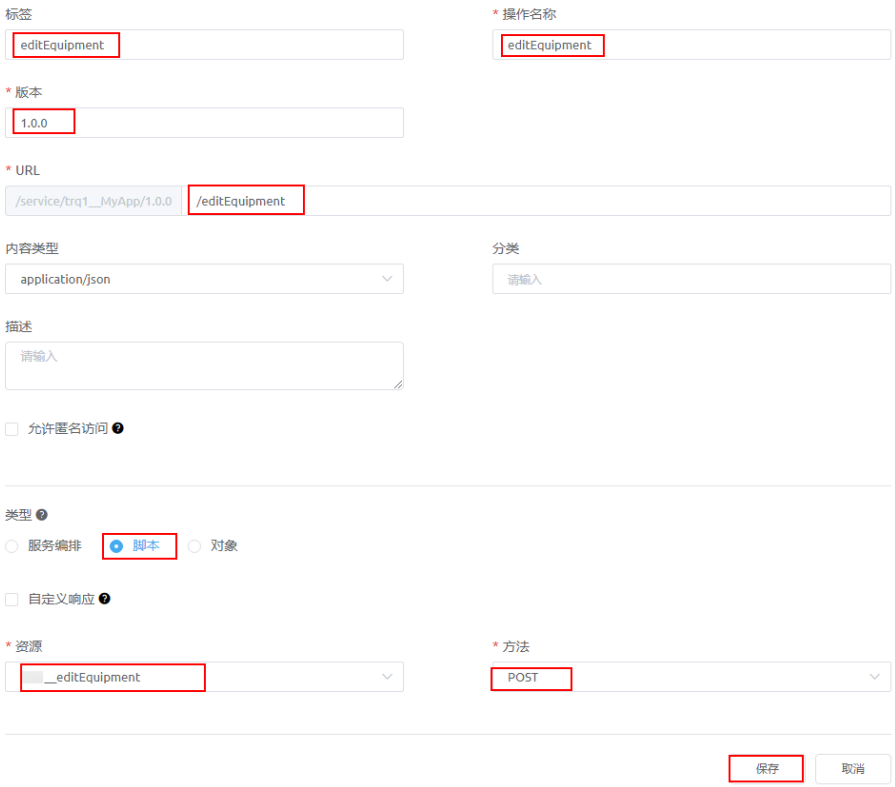
步骤3 创建“编辑设备”脚本“***HW***__editEquipment”的公共接口。

1. 设置接口参数信息，设置标签和操作名称为“editEquipment”，版本为“1.0.0”，URL为“/editEquipment”，“类型”选择“脚本”，“资源”为“***HW***__editEquipment”，方法为“POST”，然后单击“保存”。

   说明

   如果在“资源”下拉框中，未找到需要关联的脚本，请检查相关脚本是否已启用。

图 2-83 设置“编辑设备”脚本的公共接口参数



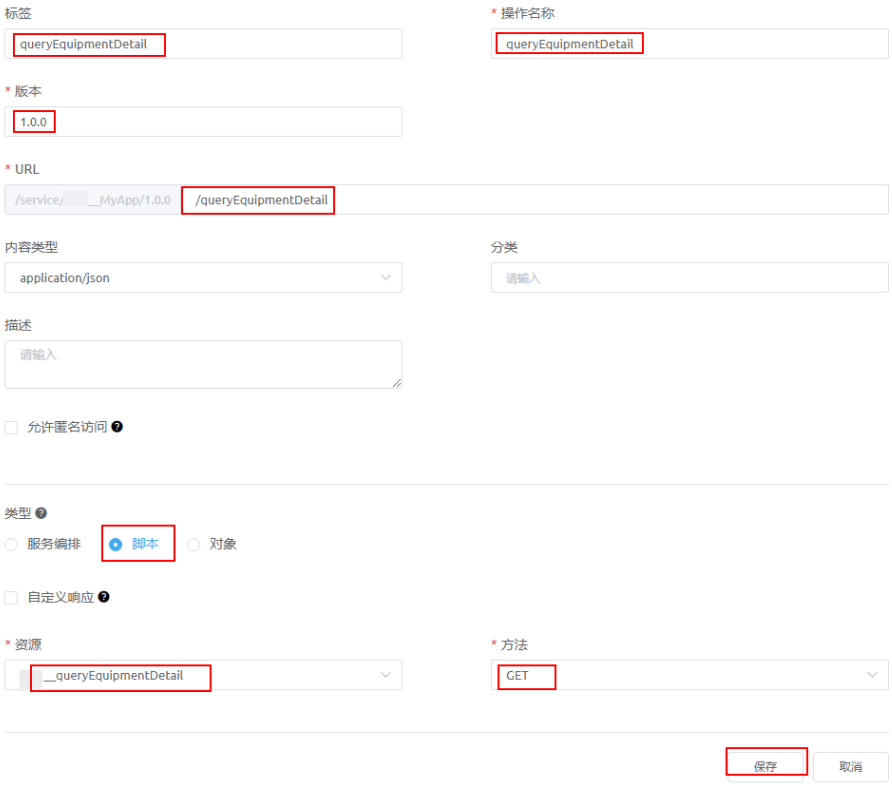
2. 在应用开发页面，单击左下角的“服务”，进入公共接口页面，查看上一步中新建的自定义接口URL“/service/***HW***__MyApp/1.0.0/editEquipment”，后续开发页面时，会使用这个URL。

图 2-84 查看自定义接口 URL

步骤4 参照上一步，创建“按ID查询设备详情”脚本“***HW***__queryEquipmentDetail”的公共接口，详细接口信息如图2-85所示。

标签和操作名称为“queryEquipmentDetail”，版本为“1.0.0”，URL为“/queryEquipmentDetail”，“类型”选择“脚本”，“资源”为“***HW***__queryEquipmentDetail”，“方法”为“GET”。

图 2-85 “*HW*__queryEquipmentDetail”的公共接口参数



----结束

2.6.3.3 组装“编辑设备”页面

组装页面包括拼装页面组件、定义组件的事件代码，并通过公共接口调用“编辑设备”脚本。

页面分析

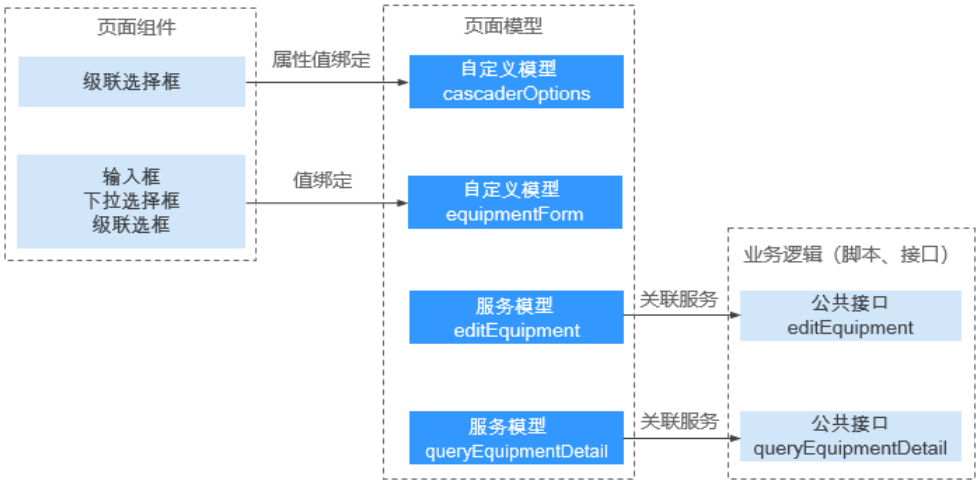
如果要实现将前端页面上输入的设备信息保存到数据库中，需要创建与前端组件绑定的自定义模型，以及与后端逻辑关联的服务模型，如图2-87所示。

图 2-86 编辑设备页面预览

页面上包含输入框（设备编码、设备名称、设备型号和地址）、选择框（设备品牌）和级联选择框（省市区）。其中，“省/市/区”级联框不仅要值绑定，还需要做属性绑定：

- 级联框的选项：以属性绑定的方式，与cascaderOptions关联。即cascaderOptions的内容，将成为级联框的可选择项。
- 级联框的选择结果：以值绑定的方式，与equipmentForm关联。即级联框的选择结果，将保持到equipmentForm.cascaderAddress中。

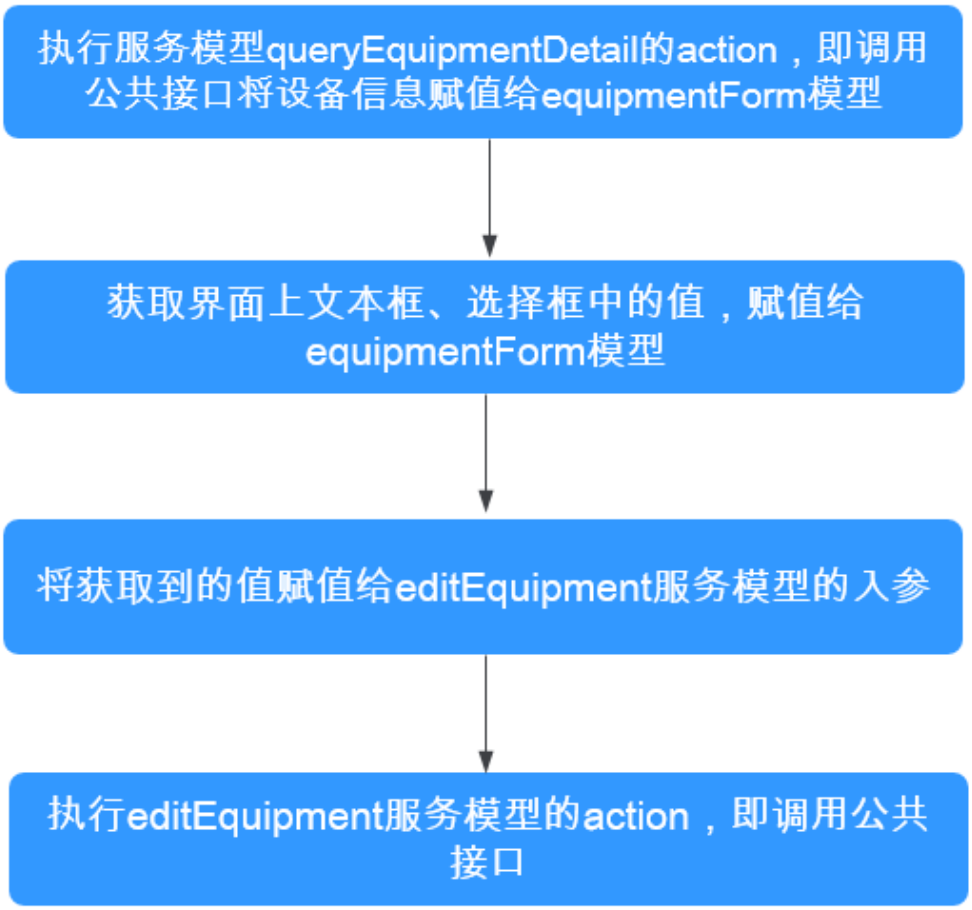
图 2-87 配置态的模型定义



同时，还需要为页面定义页面事件，为“保存”按钮定义“点击”事件，在事件的脚本代码中调用上述服务模型的API。

在应用预览下，从打开页面、在页面输入内容，到单击“保存”按钮，触发的执行逻辑如图2-88所示。

图 2-88 执行逻辑



定义模型

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”中，将鼠标放在“Page”目录上，单击界面上出现的“+”，选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“editEquipment”，单击“添加”。

平台实际创建的页面名称为“***HW***__editEquipment”，包含前缀“***HW***__”，对应首次创建应用时定义的命名空间。新建创建的页面，默认是当前用户锁定状态，可以进行编辑保存等操作。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 定义与“省/市/区”级联框的可选项相关联的自定义模型。

1. 在页面底部单击“模型视图”，进入模型视图页面，单击“新增模型”。
2. 添加自定义模型，模型名称“cascaderOptions”，单击“下一步”，如**图2-89**所示。

图 2-89 定义级联框用到的自定义模型

3. 设置保持不变，单击“下一步”。
4. 方法保持不变，单击“确定”。
5. 单击页面上方的，保存设置。

步骤5 定义与页面上各个输入框、选择框相关联的自定义模型。

1. 在“模型视图”中，单击“新增模型”。
2. 添加自定义模型，模型名称“equipmentForm”，单击“下一步”，如图2-90所示。

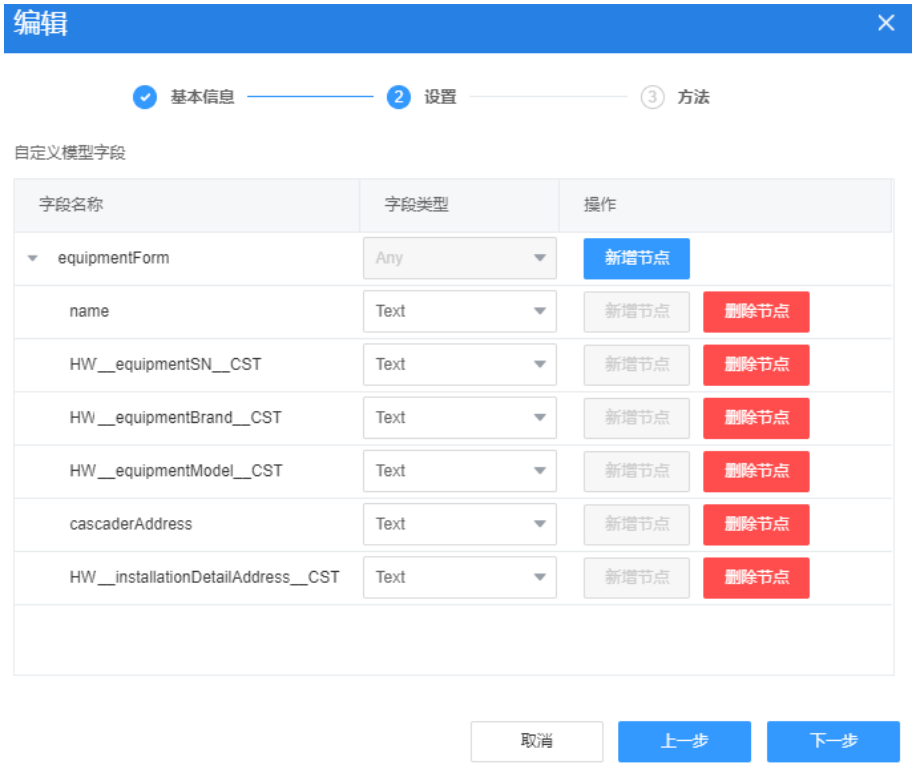
图 2-90 定义页面组件需要关联的自定义模型

3. 单击“新增节点”，逐一添加与页面元素对应的参数（name、***HW***__equipmentSN__CST、***HW***__equipmentBrand__CST、***HW***__equipmentModel__CST、cascaderAddress、***HW***__installationDetailAddress__CST），单击“下一步”，如图2-91所示。

须知

为简化后续事件脚本，除cascaderAddress外，请确保其他5个参数的参数名与设备对象（***HW***__Equipment__CST）的字段名保持一致。注意这里的下划线是两个，要与表2-5里的字段保持一致，***HW***__需要修改为实际的命名空间前缀。

图 2-91 添加模型包含的参数



4. 方法保持不变，单击“确定”。
5. 单击页面上方的，保存设置。

步骤6 定义与API（editEquipment:1.0.0）关联的服务模型。

1. 在“模型视图”中，单击“新增模型”。
2. 添加服务模型，模型名称“editEquipment”，来源选择“服务”，单击“下一步”，如图2-92所示。

图 2-92 定义服务模型

3. 指定模型与API“editEquipment”关联，单击“下一步”，如图2-93所示。
   关联API后，系统会自动显示API中脚本的输入、输出参数。

图 2-93 为模型关联 Script

4. 方法保持不变，单击“确定”。
   系统自动添加了执行的方法，如图2-94所示。未来，将在事件脚本中执行这个方法，即执行模型关联的API中的脚本。

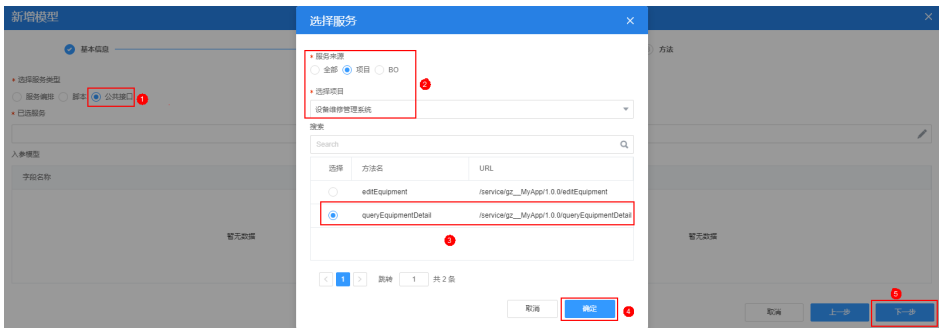
图 2-94 为模型定义方法

5. 单击页面上方的，保存设置。

步骤7 定义与API（queryEquipmentDetail）关联的服务模型。

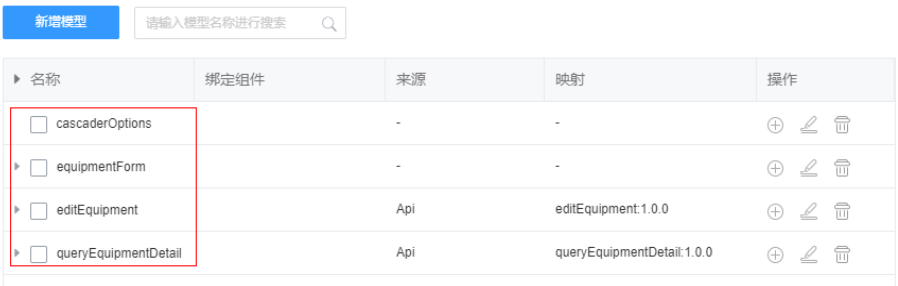
1. 在“模型视图”中，单击“新增模型”。
2. 添加服务模型，模型名称“queryEquipmentDetail”，“来源”选择“服务”，单击“下一步”。
3. 指定模型与API“queryEquipmentDetail”，单击“下一步”。

图 2-95 为模型关联 Script



4. 方法保持不变，单击“确定”。
5. 单击页面上方的，保存设置。

图 2-96 新增的页面模型

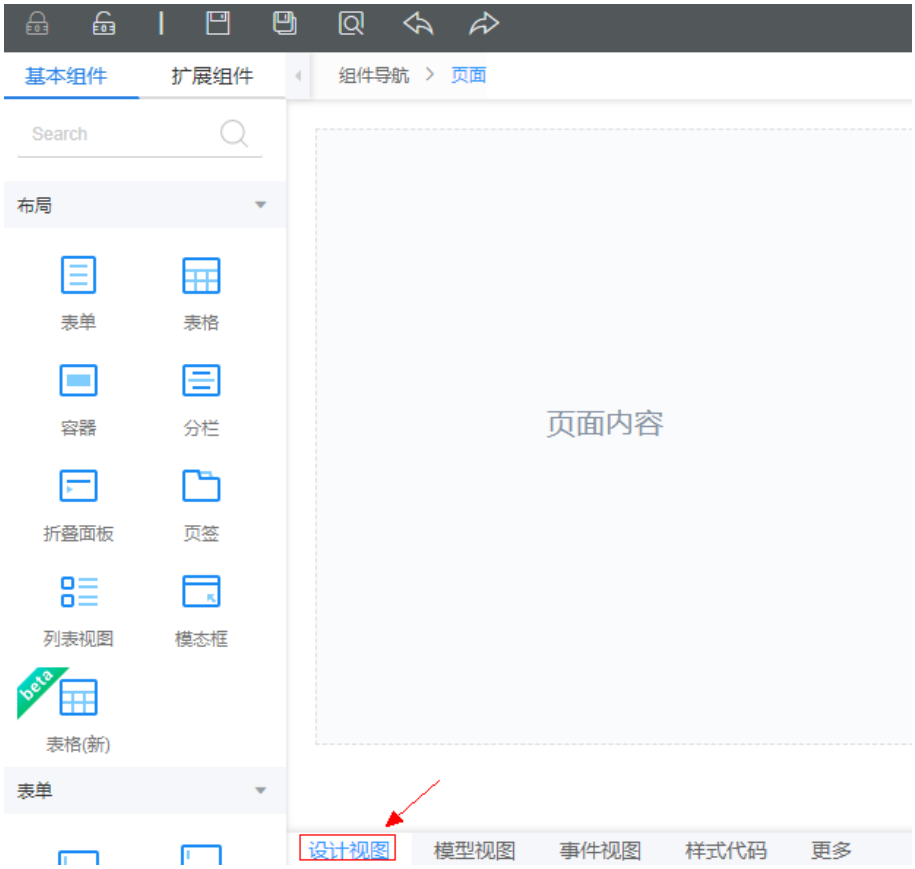


----结束

拖拽组件并关联模型

步骤1 单击“设计视图”，切换到页面设计视图。

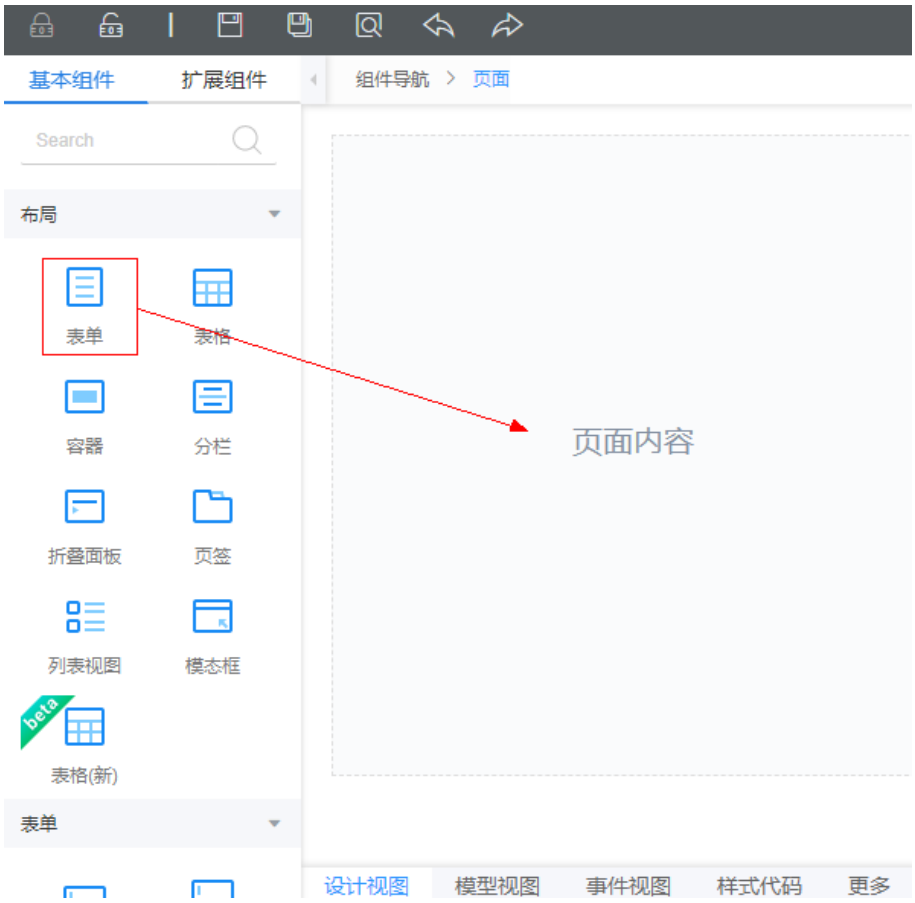
图 2-97 切换到设计视图组件列表



步骤2 将左侧组件区的“表单”拖拽到右侧“页面内容”中，在“元数据表单配置向导”弹窗底部，单击“取消”，创建一个空的表单控件。

当前不单独定义数据源，因此需要单击“取消”数据绑定。

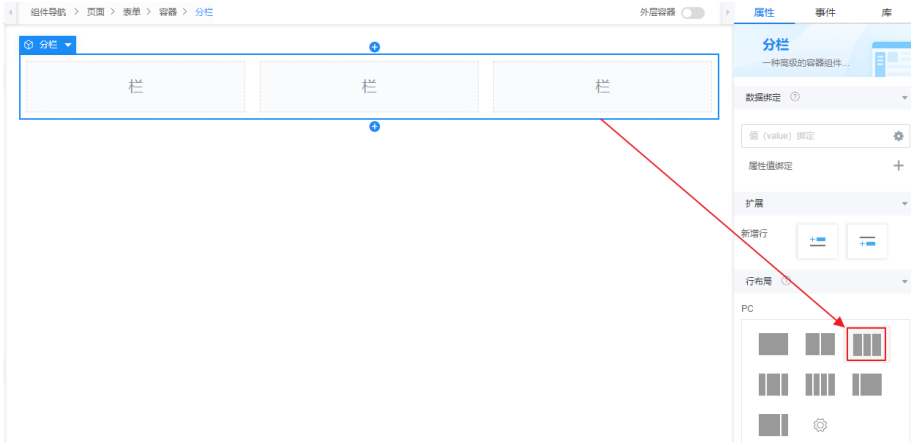
图 2-98 拖拽表单到页面并取消数据绑定



步骤3 组装参数区域。

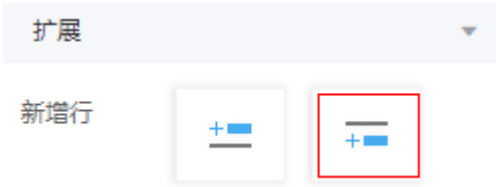
1. 从左侧组件列表中，拖一个“容器”到上一步创建的“表单”。
2. 从左侧组件列表中，拖一个“分栏”到上一步创建的“容器”中。
   “分栏”默认有2个“栏”，即当前栅格中包含1行1列的区域。
3. 选中“分栏”，在右侧属性的“行布局”中，单击，将分栏设置为3栏。

图 2-99 设置分栏为 3 栏



4. 在右侧“属性”页签中，单击“新增行”后面的图标，新增一行，如图2-100所示。设置后，分栏组件被设置为2行（分栏）3列（栏）。

图 2-100 设置表格内的行列数



5. 从左侧组件区的拖一个“输入框”到分栏组件的第1行（分栏）第1栏，并在右侧“属性”页签中将“标签”修改为“设备编码”。

图 2-101 设备编码

6. 分别向第1行第2栏、第2行第1栏、第2行第3栏中拖一个“输入框”，并设置“标签”为“设备名称”、第2行第1栏“设备型号”、第2行第3栏“详细地址”。
7. 从左侧组件列表中，拖一个“下拉框”到分栏组件的第1行第3栏，并在右侧“属性”页签中将“标签”修改为“设备品牌”。
8. 从左侧组件列表中，拖一个“级联选择框”到分栏组件的第2行第2栏，并将“标签”修改为“省/市/区”。

步骤4 组装页面标题。

在左侧组件区拖拽一个“标题”组件到上一步创建的“容器”前面，并在右侧“属性”页签中将“标题内容”修改为“设备详情”，并设置“样式”的“高级设置”为“:root{text-align:center;font-size:20px;}”。

步骤5 组装按钮区域。

1. 在左侧组件区拖拽一个“容器”到3中创建的“容器”后（**注意要在表单里面，两个容器在同级**），并在右侧“属性”页签中将“水平对齐方式”修改为“中”，即居中对齐。

图 2-102 拖拽容器

图 2-103 设置居中对齐

2. 从左侧组件区拖拽一个“按钮”到刚创建的“容器”中，并在右侧“属性”页签中，将“显示名称”修改为“保存”，将“类型”修改为“主要按钮”。
3. 拖拽一个按钮到“保存”按钮右边，并设置为“取消”按钮，类型设置为“默认按钮”。
4. 单击界面上方的  ，保存页面，可以在属性面板底部查看组件树。

图 2-104 组装完成后页面的组件树

步骤6 为页面组件关联模型。

1. 选中“设备名称”输入框。
2. 在右侧“属性”页签中单击⚙，为“设备名称”输入框绑定“equipmentForm”自定义模型中的“name”参数，如图2-105所示。

   数据绑定后，当在前台界面输入内容时，系统就会把输入框中的内容，赋值给“name”。

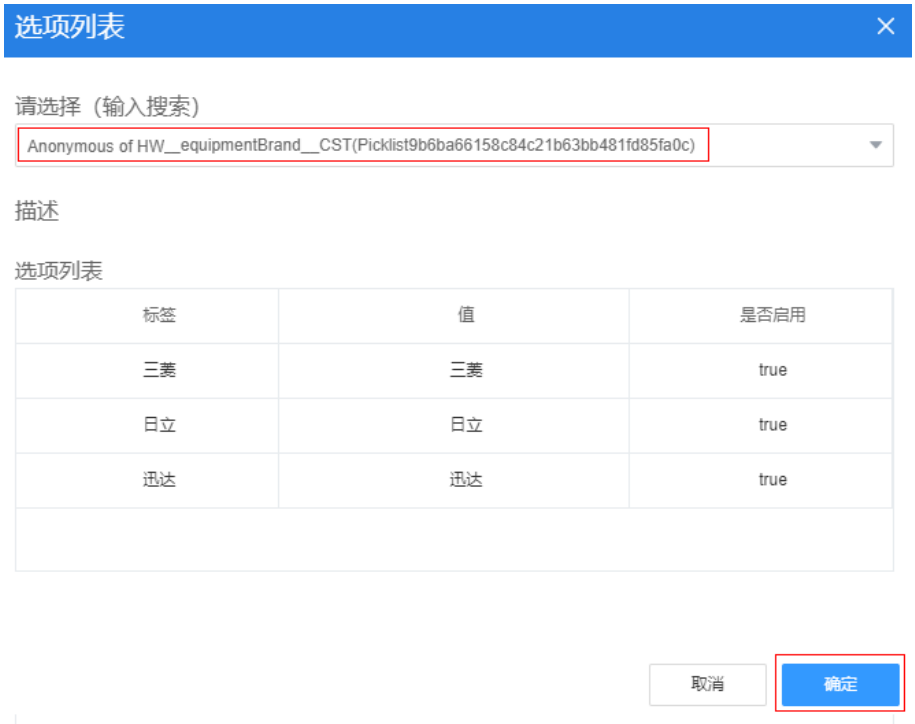
图 2-105 输入框数据绑定

3. 重复上一步，为“设备编码”、“设备型号”、“详细地址”文本输入框绑定“equipmentForm”自定义模型中的对应参数。
4. 选中“设备品牌”下拉选择框，在右侧“属性”页签中单击，为选择框绑定“equipmentForm”自定义模型中的“HW__equipmentBrand__CST”参数。
5. 选中“设备品牌”下拉选择框，在右侧“基本属性”中，单击“选项列表”的，在弹窗中输入“equipmentBrand”，在联想记录中，选择设备对象字段“*HW*__equipmentBrand__CST”，为选择框添加下拉选项关联的字段，如图2-106所示。

图 2-106 为下拉选择框定义可选项

图 2-107 输入字段搜索

图 2-108 关联字段



6. 选中“省/市/区”级联框，在右侧“属性”页签中单击，为级联框绑定“equipmentForm”自定义模型中的“cascaderAddress”参数。
7. 单击属性值绑定后的“+”，将“属性”设置为“选择数据”，“模型字段”绑定到“cascaderOptions”，如图2-109所示。

   数据绑定后，级联选择框的选项来自“cascaderOptions”；同时，当在前台界面选择省/市/区时，系统就会把选择结果赋值给“equipmentForm.cascaderAddress”。

图 2-109 级联框数据绑定

8. 单击页面上方的，保存页面。

----结束

定义页面事件

步骤1 定义页面加载事件。

1. 在“设计视图”中，选中最外层的“页面”，也可以直接单击组件导航，快速定位。
2. 在右侧“事件”页签中，单击“加载”后的“+”，为页面添加事件代码。

图 2-110 添加页面代码

3. 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

```
//debugger;
//为级联组件下拉框赋值
$model.ref('cascaderOptions').setData([{"value":"110000000000","label":"北京市","children":
[{"value":"110100000000","label":"市辖区","children":[{"value":"110101000000","label":"东城区"},
{"value":"110102000000","label":"西城区"},{"value":"110105000000","label":"朝阳区"},
{"value":"110108000000","label":"海淀区"}]}]},{"value":"310000000000","label":"上海市","children":
[{"value":"310100000000","label":"市辖区","children":[{"value":"310101000000","label":"黄浦区"},
{"value":"310104000000","label":"徐汇区"},{"value":"310106000000","label":"静安区"},
{"value":"310115000000","label":"浦东新区"},{"value":"310110000000","label":"杨浦区"},
{"value":"310112000000","label":"闵行区"}]}]},{"value":"440000000000","label":"广东省","children":
[{"value":"440100000000","label":"广州市","children":[{"value":"440101000000","label":"市辖区"},
{"value":"440105000000","label":"海珠区"},{"value":"440106000000","label":"天河区"},
{"value":"440111000000","label":"白云区"},{"value":"440113000000","label":"番禺区"}]},
{"value":"440300000000","label":"深圳市","children":[{"value":"440301000000","label":"市辖区"},
{"value":"440303000000","label":"罗湖区"},{"value":"440304000000","label":"福田区"},
{"value":"440305000000","label":"南山区"},{"value":"440306000000","label":"宝安区"},
{"value":"440307000000","label":"龙岗区"},{"value":"440308000000","label":"盐田区"},
{"value":"440309000000","label":"龙华区"},{"value":"440310000000","label":"坪山区"}]}]}]);
//从页面url中获取id，然后将id作为queryEquipmentDatail的入参
let id =Page.params.id;
if(id && id !=""){
let _model = $model.ref("queryEquipmentDetail");
    _model.setValue("inputParam",{"equipmentId": id });
    _model.run().then(()=>{
//获取queryEquipmentDetail的出参后赋值给页面表单模型equipmentForm
var data = _model.getData();
if(data.outputParam && data.outputParam.equipment){
var equip = data.outputParam.equipment;
            $model.ref("equipmentForm").setData(equip);
}
}).catch((e)=>{
this.$dialog.error({
            title:'错误',
            content: e.resMsg
})
});
}
```

4. 单击“创建”，关闭事件编排器，返回到页面。

步骤2 定义“保存”按钮的“点击”（on-click）事件。

1. 在“设计视图”中，选中“保存”按钮。
2. 在右侧“事件”页签中，单击“点击”后的+。
3. 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名、页面名。

```
//debugger;
// 当前组件
let _component = context.$component.current;
//校验表单
_component.getForm().formValidate().then((val) => {
    //校验成功
    //从自定义模型equipmentForm中获取界面表单数据
    let data = $model.ref('equipmentForm').getData();
    if (data) {
        console.log(data);
        //构造Script（editEquipment）的其他入参
        let inputs = data;
        inputs.HW__installationSiteProvince__CST = "";//构造参数名称要与Object字段名保持一致
        inputs.HW__installationSiteCity__CST = "";
        inputs.HW__installationSiteArea__CST = "";
        inputs.HW__fullAddress__CST = "";
        inputs.HW__longitude__CST = "";
        inputs.HW__latitude__CST = "";
        //获取服务模型editEquipment，执行模型的action，即执行模型关联的Script
        let _model = $model.ref('editEquipment');
        if (!inputs.cascaderAddress || inputs.cascaderAddress.length < 1) {
            _model.setValue('inputParam', { "equipment": inputs });
            _model.run().then(() => {
                //提交成功后返回设备列表页面
                context.$page.load('/besBaas/page#/HW__equipmentManage');
            }).catch((e) => {
                this.$dialog.error({
                    title: '错误',
                    content: e.resMsg
                })
            });
        }
        else {
            console.log('data');
            //将级联框的内容转换为Script（editEquipment）的省、市、区3个参数，并拼接完整地址
            let fullAddress = "";
            let cascaderOptions = $model.ref('cascaderOptions').getData();
            if (cascaderOptions.length < 1) {
                throw new Error("CascaderOptions data is wrong.");
            }
            cascaderOptions.forEach(function (province, idx) {
                if (province.value == inputs.cascaderAddress[0]) {
                    //构造参数名称要与Object字段名保持一致
                    inputs.HW__installationSiteProvince__CST = inputs.cascaderAddress[0];
                    fullAddress += province.label;
                    if (inputs.cascaderAddress.length > 1) {
                        province.children.forEach(function (city, idxCity) {
                            if (city.value == inputs.cascaderAddress[1]) {
                                //构造参数名称要与Object字段名保持一致
                                inputs.HW__installationSiteCity__CST = inputs.cascaderAddress[1];
                                fullAddress += city.label;
                                if (inputs.cascaderAddress.length > 2) {
                                    city.children.forEach(function (area, idxArea) {
```



```
                                if (area.value == inputs.cascaderAddress[2]) {
                                    //构造参数名称要与Object字段名保持一致
                                    inputs.HW__installationSiteArea__CST = inputs.cascaderAddress[2];
                                    fullAddress += area.label;
                                    return false;
                                }
                            });
                        }
                        return false;
                    }
                });
            }
            return false;
        }
    });
    //界面上输入了详细地址时，拼接出fullAddress完整地址
    if (inputs.HW__installationDetailAddress__CST) {
        fullAddress += inputs.HW__installationDetailAddress__CST;
    }
    if (fullAddress) {
        console.log('fullAddress');
        inputs.HW__fullAddress__CST = fullAddress;
        _model.setValue('inputParam', { "equipment": inputs });
        _model.run().then(() => {
            context.$page.load('/besBaas/page#/HW__equipmentManage');
        }).catch((e) => {
            this.$dialog.error({
                title: '错误',
                content: e.resMsg
            })
        });
    }
    else {
        _model.setValue('inputParam', { "equipment": inputs });
        _model.run().then(() => {
            //提交成功后返回设备列表页面
            context.$page.load('/besBaas/page#/HW__equipmentManage');
        }).catch((e) => {
            this.$dialog.error({
                title: '错误',
                content: e.resMsg
            })
        });
    }
  }
}).catch((error) => {
});
```

4. 单击“创建”，关闭事件编排器，返回到页面。

步骤3 定义“取消”按钮的“点击”（on-click）事件。

1. 在“设计视图”中，选中“取消”按钮。
2. 在右侧“事件”页签中，单击“点击”后的“+”。
3. 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

```
//返回设备列表页面
context.$page.load('/besBaas/page#/HW__equipmentManage');
```

4. 单击“创建”，关闭事件编排器，返回到页面。
5. 单击页面上方的  ，保存页面。

----结束

验证

在页面预览中，检查页面效果。

步骤1 单击界面上方的  ，进入预览应用页面。

系统会弹出“***HW***__editEquipment”预览页面。

步骤2 检查“设备品牌”下拉框，“省市区”级联框的选项是否正确。

选项正确，则说明自定义模型、页面组件与模型绑定关系是正确的。因为当前编辑完成后的保存跳转页面还没有创建，所以此时“保存”、“取消”按钮，还不能验证。

----结束

2.6.4 开发“管理设备”功能

2.6.4.1 背景及原理（服务编排）

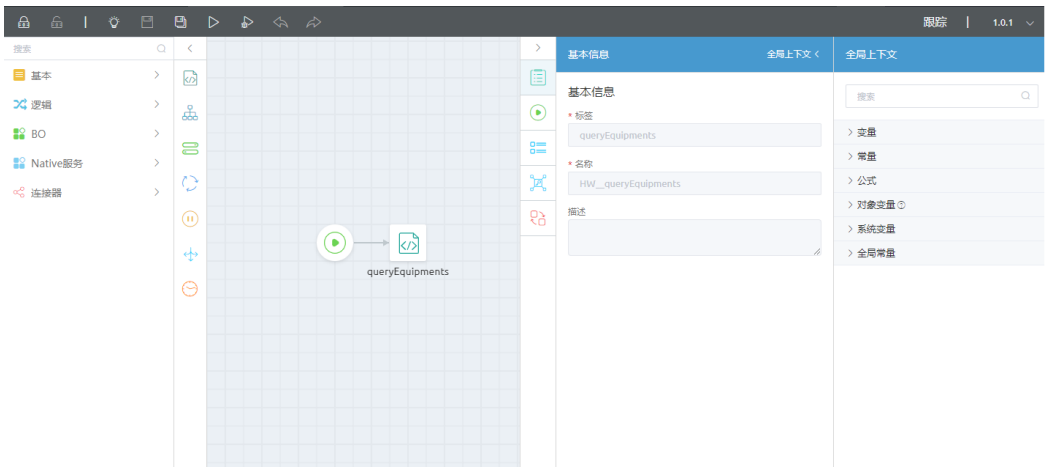
AstroZero的服务编排，支持对逻辑判断组件、数据处理组件，以及脚本、子服务编排、商业对象等进行可视化组合编排，实现丰富的业务功能。

了解服务编排

在传统的开发中程序员一般是基于代码进行开发，程序员需要学习内容较多，开发效率相对低一些，开发门槛也高。AstroZero的服务编排功能，类似于编程中一段有流程、条件处理、判断逻辑的程序。这段程序有输入参数和输出参数、可以独立成为一个对外调用的方法。同时，在程序内部，也可以调用其他的方法。

AstroZero中的服务编排是将原来基于代码编程改变为用图形化，拖拉拽的方式去编程。如图2-111所示，服务编排界面是图形化、模板化的，您甚至不需要任何编程经验，将左侧面板区的组件拖拽到右侧画布、做必要的配置，就可以完成服务编排的开发。

图 2-111 服务编排界面



服务编排界面中，可以编排如下组件：

- 基本组件：在服务编排引用脚本或者另一个服务编排，增/改/删/查记录，以及发送邮件、发送事件等。
- 逻辑组件：在服务编排中实现变量赋值Assignment、循环Loop、跳出循环Break、决策Decision和等待Wait。

- 商业对象：将封装好的BO能力作为服务编排中的一个节点。
- 连接器：将短信发送、支付等第3方连接器作为当前服务编排中的一个节点。

除了图形化编排，AstroZero也支持服务编排的在线测试验证，以及问题跟踪调试，方便您及时发现并解决问题。

服务编排测试通过、发布后，既可以直接被前端页面调用，也可以作为restful接口被第三方系统调用，也可以包装成公共接口后被调用。本节中主要是将服务编排包装成一个公共接口后，供页面调用，“管理设备”功能中涉及的业务逻辑，以及服务编排与脚本关系如**表2-8**下所示，详细操作方式及说明请参见**创建业务逻辑**。

表 2-8 “管理设备”功能需要创建的脚本、服务编排详情

脚本名称	主要作用	关联服务编排	关联公共接口
HW__queryEquipments	在输入查询条件后，查询设备	***HW***__queryEquipments	queryEquipments
HW__deleteEquipment	删除设备	不涉及	deleteEquipment
HW__equipmentSelectListQuery	查询所有设备并以选项列表的形式返回	不涉及	equipmentSelectListQuery

2.6.4.2 创建业务逻辑

2.6.4.2.1 创建“查询设备”脚本

管理设备页面中，实现输入设备的信息，查询出对应设备的全部信息功能，需要开发“查询设备”脚本和服务编排。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”目录中，将鼠标放在“Script”上，单击界面上出现的  ，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“queryEquipments”，单击“添加”。

系统实际创建的脚本名称为“***HW***__queryEquipments”，“***HW***__”前缀由租户命名空间namespace决定。新建创建的脚本，默认是当前用户锁定状态，可以进行编辑保存等操作。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 在代码编辑器中，插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
/*****************************
 * 本脚本用于按条件查询设备列表
 * *************************/
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import * as decimal from 'decimal';//导入decimal数据类型相关的标准库
//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String' })
    name: string;//设备名称，需要按设备名称查询时传入
    @action.param({ type: 'String' })
    fullAddress: string;//完整地址，需要按完整地址查询时传入
    @action.param({ type: 'Number', min: 0 })
    start: decimal.Decimal;//分页信息，表示从第几条数据开始查询
    @action.param({ type: 'Number', min: 0 })
    limit: decimal.Decimal;//分页信息，表示一次查询几条数据
}
//定义出参结构
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any', isCollection: true, label: 'object' })
    equipments: object[];//设备列表
    @action.param({ type: 'String' })
    total: string;//总共查到几条数据
}
@useObject(['HW__Equipment__CST']) //使用数据库对象HW__Equipment__CST
@action.object({ type: "method" })
export class QueryEquipments {
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public queryEquipments(input: ActionInput): ActionOutput {
        let out = new ActionOutput(); //新建出参ActionOutput类型的实例，作为返回值
        try {
            //获取HW__Equipment__CST这个Object的操作实例
            let s = db.object('HW__Equipment__CST');
            //option是db标准库queryByCondition方法的入参（选项选项），用于对查询结果排序、分组、聚合计
算等
            //这里构造的是按创建时间逆序返回查询结果
            let option = {
                "options": {
                    "orderby": [
                        {
                            "field": "createdDate",
                            "order": "desc"
                        }
                    ],
                }
            };
            //如果有分页
            if (input.start && input.limit) {
                let start = decimal.toNumber(input.start);//将decimal类型转换为接口需要的number类型
                let limit = decimal.toNumber(input.limit);
                option.options['limit'] = limit;
                option.options['skip'] = start;
            }
            //condition是db标准库queryByCondition方法的入参（查询条件）
            let condition = {
                "conjunction": "AND",
                "conditions": []
            };
            //基本查询条件
            condition.conditions.push({
                "field": "id",
                "operator": "isnotnull",
            });
            condition.conditions.push({
                "field": "id",
                "operator": "ne",
                "value": ""
```

```
        });
        //按设备名称查询
        if (input.name && input.name != "") {
            condition.conditions.push({
                "field": "name",//与对象中的字段名保持一致
                "operator": "contains",
                "value": input.name
            });
        }
        //按设备地址查询
        if (input.fullAddress && input.fullAddress != "") {
            condition.conditions.push({
                "field": "HW__fullAddress__CST",
                "operator": "contains",
                "value": input.fullAddress
            });
        }
        //调用按条件查询HW__Equipment__CST的接口
        out.equipments = s.queryByCondition(condition, option);
        //构造实时监控页面需要的属性字段longitude__CST、latitude__CST、fullAddress__CST和
equipmentSn__CST
        for (let equip of out.equipments || []) {
            equip['longitude__CST'] = equip['HW__longitude__CST'];//与对象中的字段名保持一致
            equip['latitude__CST'] = equip['HW__latitude__CST'];
            equip['fullAddress__CST'] = equip['HW__fullAddress__CST'];
            equip['equipmentSn__CST'] = equip['HW__equipmentSN__CST'];
        }
        //调用查询符合condition条件的数据总数的接口
        out.total = s.count(condition) + "";
    } catch (error) {
        console.error(error.name, error.message);
        context.setError(error.name, error.message);
    }
    return out;
  }
}
```

步骤5 单击编辑器上方的，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的，执行脚本。
2. 在界面底部，直接单击测试窗口右上角，执行脚本。
   执行成功后，会在“输出”页签返回全部设备信息。

步骤7 测试成功，单击编辑器上方的，启用脚本。

----结束

2.6.4.2.2 创建“查询设备”服务编排

“查询设备”脚本创建完成后，创建一个“服务编排”，并在服务编排中引用脚本，改造和配置服务编排的输入输出参数，使其更适用于前端页面调用。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”目录中，将鼠标放在“Flow”上，单击界面上出现的，在弹出菜单中选择“服务编排”。

步骤3 选中“创建一个新的服务编排”，在“标签”和“名称”文本框中输入“queryEquipments”，并设置类型为“Autolaunched Flow”，单击“添加”。

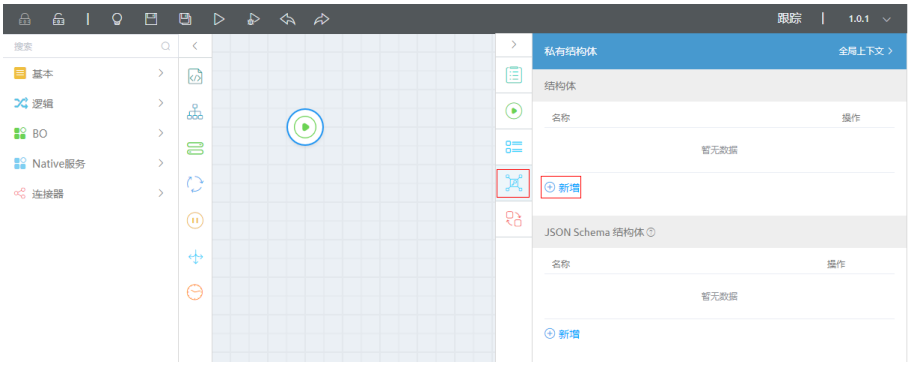
系统实际创建的脚本名称为“***HW***__queryEquipments”，“***HW***__”前缀由租户命名空间namespace决定。新建创建的服务编排，默认是当前用户锁定状态，可以进行编辑保存等操作。

当编辑已有服务编排时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 定义服务编排用到的变量。

1. 单击页面右侧的，再单击结构体中的“新增”，在弹出窗口中输入结构体类型名称“Equipment”，单击“保存”。

图 2-112 创建私有结构体



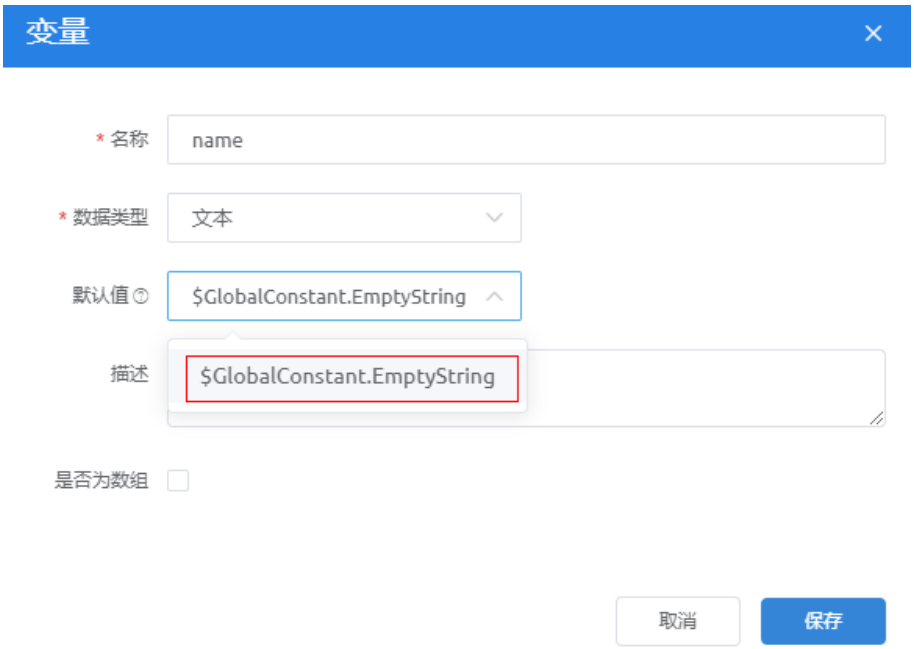
2. 单击，再单击“对象变量”后的“+”，在弹出窗口中定义私有结构体变量“equipments”，选中“是否为数组”（该变量将作为服务编排的输出参数，且因为查询结果可能是多条记录，所以要选）。单击“保存”，如图2-113所示。

图 2-113 新增私有结构体变量

3. 单击“变量”后的“+”，设置参数名称为“name”。
   单击变量后的“...”，选择“设置”，修改变量的名称、类型，选择默认值“{!$GlobalConstant.EmptyString}”信息，如图2-115所示。

图 2-114 新增变量

图 2-115 设置变量



4. 重复上一步，定义**表2-9**中的其他变量。

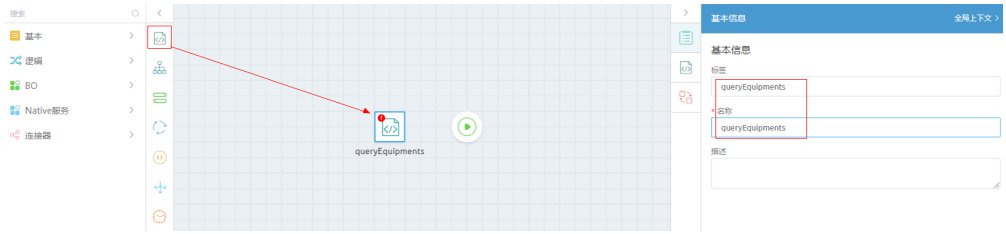
表 2-9 服务编排变量说明

变量唯一标识	描述	数据类型	默认值
name（上一步已定义）	设备名称	文本	{! $GlobalConstant.EmptyString}
fullAddress	完整地址	文本	{! $GlobalConstant.EmptyString}
limit	分页查询的条目数	数字	{!$GlobalConstant.Null}
start	分页查询开始条目	数字	{!$GlobalConstant.Null}
total	查询到的总条目数	文本	不设置

步骤5 在服务编排中添加并配置脚本图元。

1. 从左侧拖拽脚本图元到画布中。
2. 修改图元的基本信息，“标签”和“名称”都设置为“queryEquipments”。

图 2-116 设置脚本图元基本信息



3. 单击  ，指定图元对应的脚本名称（***HW***__queryEquipments），并配置脚本的输入输出参数，如果下拉选项中没有目标脚本，请检查脚本是否已启用。

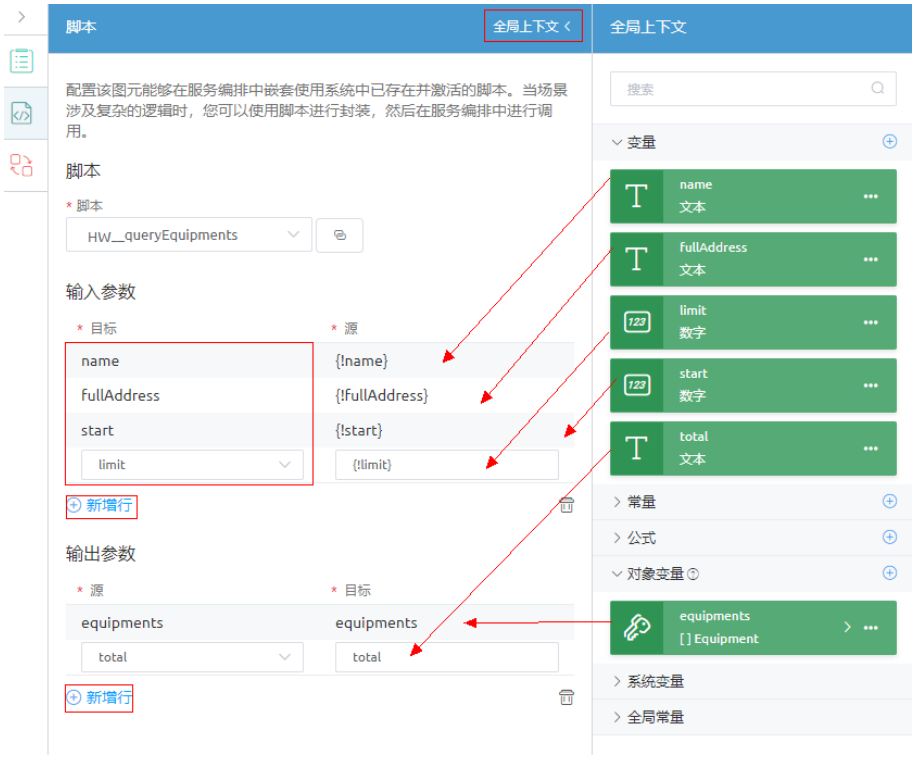
   单击“全局上下文”，显示变量列表。在“输入参数”中，单击“新增行”，在下拉框中选择输入参数、依次**拖拽**变量到输入参数的“源”下。

   在“输出参数”中，单击“新增行”，在下拉框中选择输出参数，并从变量和对象变量中拖拽“目标”，如图2-117所示。

   说明

   请直接从全局上下文拖拽变量到输入输出参数下的对应位置，请勿手动输入，手动输入的值系统可能不识别。

图 2-117 拖拽脚本的输入输出参数



步骤6 定义服务编排的输入、输出参数，并保存服务编排。

1. 在画布上，把鼠标放在起点图元  图元上，从“+”拖动鼠标，在起点图元和脚本图元间增加连线；即将当前脚本设置为服务编排的起始节点。
2. 鼠标在画布空白处点一下，单击右侧  ，设置服务编排的输入输出参数，如图2-118所示。

图 2-118 拖拽服务编排的输入输出参数

3. 单击服务编排界面上方的，保存服务编排。

步骤7 测试服务编排能否正常执行。

1. 单击服务编排编辑器上方的，执行服务编排。
2. 在“输入参数”中，输入测试数据，单击“运行”。
   执行成功，界面上会返回设备对象中的全部信息，样例如下：

图 2-119 返回样例

步骤8 （可选）在服务编排编辑器单击“跟踪”，可以查看到上一步的执行日志，方便定位错误。

步骤9 测试成功，单击服务编排编辑器上方的  ，发布服务编排。

----结束

2.6.4.2.3 创建“删除设备”脚本

当用户操作“删除”图标删除某条设备记录时，需要根据Id在设备对象中删除设备，因此需要创建一个根据Id“删除设备”的脚本。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 选中“创建一个新脚本”，在“名称”文本框中输入“deleteEquipment”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于删除设备
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
//定义入参结构，入参包含1个参数：Equipment对象，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    id: string;
}
//定义出参结构，出参包含1个参数，Equipment的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
}
//使用数据对象HW__Equipment__CST
@useObject(['HW__Equipment__CST'])
@action.object({ type: "method" })
export class DeleteEquipment {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public deleteEquipment(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let id = input.id;
            let s = db.object('HW__Equipment__CST');    //获取HW__Equipment__CST这个Object的操作实例
            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "id",
                    "operator": "eq",
                    "value": id
                }]
            };
            let isDeleted = s.deleteByCondition(condition);
            if (isDeleted) {
                out.id = id;
            } else {
                error.name = "EQERROR";
                error.message = "删除设备失败！";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            error.Error(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击脚本编辑器上方的，保存脚本。

----结束

验证

步骤1 单击脚本编辑器上方的，执行脚本。

步骤2 在界面底部单击测试窗口右上角执行图标，进行测试。

如果设置输入参数（id来自的**编辑设备脚本**测试结果）：

```
{
    "id": "cQue000000e1qnhgtCng"
}
```

则脚本返回设备的详细信息，样例如下。

```
{
    "id": "cQue000000e1qnhgtCng"
}
```

步骤3 测试成功，单击编辑器上方  启用图标，发布脚本。

----结束

2.6.4.2.4 创建“查询设备列表”脚本

查询设备列表脚本是为了实现将查询的所有设备并以选项列表的形式返回，并通过页面调用将查询结果展示到页面上，此脚本将会在**开发“生成工单”功能**章节被调用。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 选中“创建一个新脚本”，在“名称”文本框中输入“equipmentSelectListQuery”，单击“添加”。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
import * as context from 'context';
import * as decimal from 'decimal';
import * as db from 'db';
//使用数据库对象HW__Equipment__CST
@useObject(['HW__Equipment__CST'])
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String' })
    id: string;
}
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any', label: 'object', isCollection: true })
    equipList: object[];
    @action.param({ type: 'Any', label: 'object' })
    equipment: object;
}
@action.object({ type: "method" })
export class EquipmentSelectListQuery {
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public equipmentSelectListQuery(input: ActionInput): ActionOutput {
        let out = new ActionOutput();
        let error = new Error;
        try {
            let objAct = db.object('HW__Equipment__CST');
```

```
        if (input.id) {
            out.equipment = objAct.query(input.id);
            return out;
        }
        let option = {
            "options": {
                "orderby": [
                    {
                        "field": "lastModifiedDate",
                        "order": "desc"
                    }
                ],
                "skip": 0,
                "limit": 1000
            }
        };
        let equipList = objAct.queryByCondition(null, option);
        let selectValue = equipList.map(function (v, i, a) {
            return {
                'value': v['id'],
                'display': v['name']
            }
        });
        out.equipList = selectValue;
    } catch (error) {
        console.error(error.name, error.message);
        context.setError(error.name, error.message);
    }
    return out;
  }
}
```

步骤5 单击脚本编辑器上方的，保存脚本。

----结束

验证

步骤1 单击脚本编辑器上方的，执行脚本。

步骤2 在界面底部单击测试窗口右上角执行图标，进行测试。

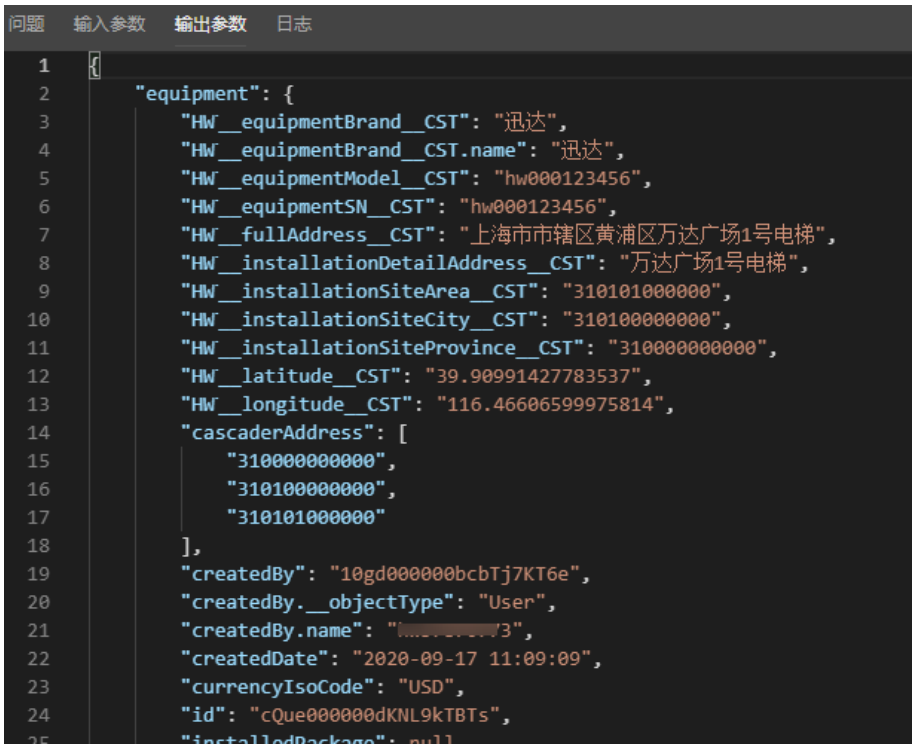
如果不设置输入参数，会出现类似如下样例的测试结果（设备名称和设备id）。

```
{
  "equipList": [
    {
      "display": "百草园B栋2单元2号",
      "value": "cQue000000e1qnhgtCng"
    }
  ]
}
```

如果设置如下输入参数。

```
{
  "id":"cQue000000e1qnhgtCng"
}
```

则脚本返回设备的详细信息，样例如下。



步骤3 测试成功，单击编辑器上方的，发布脚本。

----结束

2.6.4.2.5 创建公共接口

创建与脚本“***HW***__deleteEquipment”、“***HW***__equipmentSelectListQuery”以及服务编排“***HW***__queryEquipments”一一对应的公共接口，让页面直接调用这种公共接口。

由于脚本“***HW***__queryEquipments”已被同名的服务编排给封装引用，因此，只需要为服务编排“***HW***__queryEquipments”创建一个关联的公共接口即可，脚本则不需要重复创建。

操作步骤

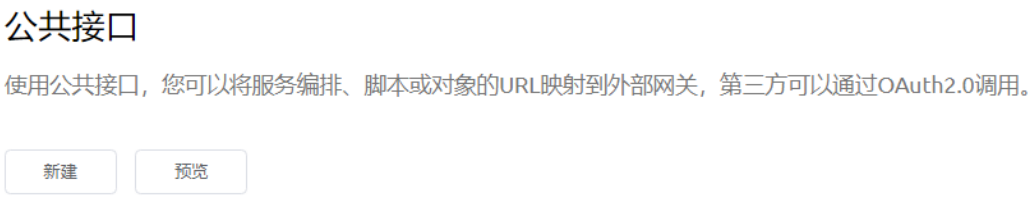
步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 单击页面下方的“服务”，进入公共接口创建页面。

图 2-120 创建公共接口入口

步骤3 单击“新建”，进入公共接口创建页面。

图 2-121 公共接口创建



步骤4 创建“**删除设备**”脚本对应的“***HW***__deleteEquipment”的公共接口。

设置接口参数信息，设置操作名称为“deleteEquipment”，版本为“1.0.0”，URL为“/deleteEquipment”，“类型”选择“脚本”，“资源”为“***HW***__deleteEquipment”，方法为“DELETE”，然后单击“保存”。

说明

如果在“资源”下拉框中，未找到需要关联的脚本或服务编排，请检查相关脚本和服务编排是否已启用。

步骤5 参照上一步，创建表2-10中其他的公共接口，详细接口信息如表2-10所示。

说明

“HW__”请修改为实际命名空间前缀。

表 2-10 公共接口

设置操作	版本	URL	方法	类型	资源
deleteEquipment（上一步已创建）	1.0.0	/deleteEquipment	DELETE	脚本	***HW***__deleteEquipment

设置操作	版本	URL	方法	类型	资源
equipmentSelectListQuery	1.0.0	/equipmentSelectListQuery	GET	脚本	***HW***__equipmentSelectListQuery
queryEquipments	1.0.0	/queryEquipments	POST	服务编排	***HW***__queryEquipments

----结束

2.6.4.3 组装“设备管理”页面

通过组装设备管理的标准页面，实现对设备对象的增、删、改、查处理。

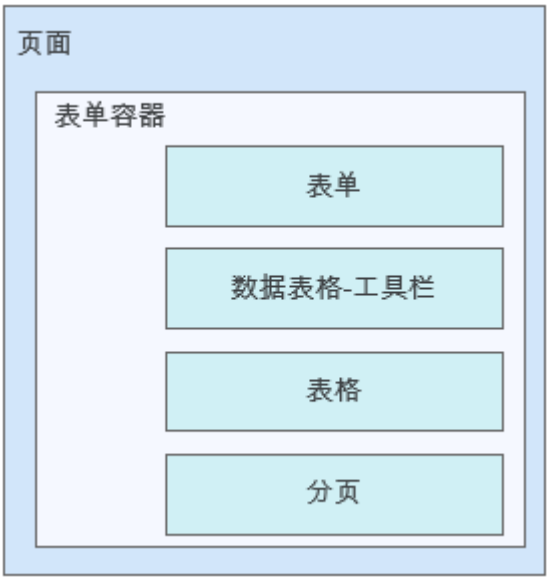
页面分析

管理设备页面需要实现以下功能，如图2-122所示：

- 工具栏保留“新增设备”按钮，单击“新增设备”按钮，跳转到设备详情页面。
- 查询结果增加操作列，包含编辑和删除图标。

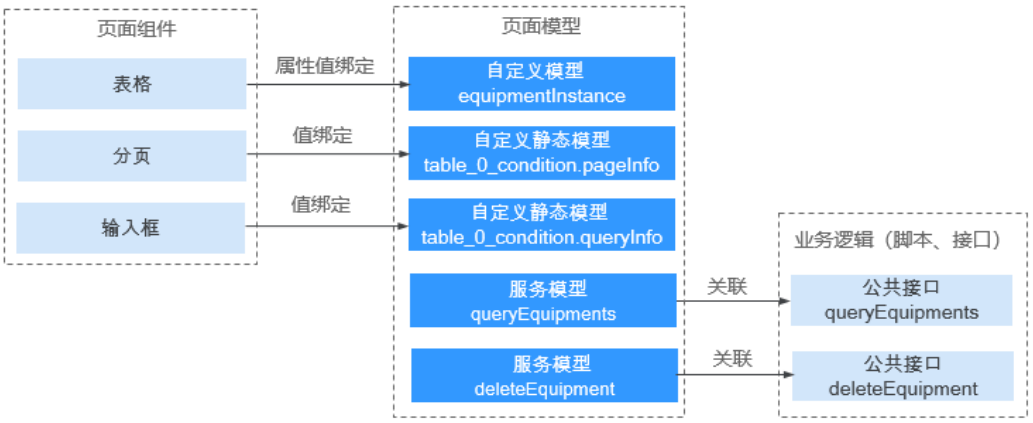
图 2-122 电梯信息管理界面

图 2-123 前端页面组件位置关系



如果想要实现将前端页面上输入的设备信息保存到数据库中，需要将页面组件、模型、与数据对象进行关联，其关联关系如图2-124所示。

图 2-124 组件、模型、与对象的关系分析



开发页面的大致流程

1. 拖拽“表格”组件到页面上。
   拖拽后，“表格”组件内默认包含“表格”和“分页”2个下层组件。
2. 创建自定义模型“equipmentInstance”，以及子节点，后续将会通过脚本（修改、查询、删除）操作对象“HW__Equipment__CST”。
3. 为“表格”绑定自定义模型“equipmentInstance”。
4. 为“表格”增加工具栏。
   配置过程中，平台会自动创建自定义模型“table_0_condition”，在自定义模型“table_0_condition”下创建查询子节点“queryinfo”，后续将会与查询条件输入框、分页组件绑定。

执行流程

完成上述配置后，一个简易的增删改查电梯设备信息的页面即开发完成，页面的执行流程如表2-11所示。

表 2-11 执行流程

场景	系统执行流程
查询	1. 界面查询条件传入自定义模型“table_0_condition”。 2. 将自定义模型“table_0_condition”的数据作为入参，运行服务模型“queryEquipments”（绑定“queryEquipmentes”接口），执行接口绑定的“***HW***__queryEquipments”脚本，从“***HW***__Equipment__CST”对象中获取设备信息列表。 3. 将服务模型“queryEquipments”返回的参数值绑定在自定义模型“equipmentInstance”并渲染“表格”。
新增	1. “表格”中填写新设备记录后，即将填写内容传入表格绑定的自定义模型“equipmentForm”。 2. 将自定义模型“equipmentForm”的数据作为入参，运行服务模型“editEquipment”（绑定“editEquipment”接口），执行接口绑定的“***HW***__editEquipment”脚本，将新增记录插入到对象“***HW***__Equipment__CST”中。
修改	1. “表格”中修改已有设备记录后，将修改内容传入表格绑定的自定义模型“equipmentForm”。 2. 将自定义模型“equipmentForm”的数据作为入参，运行服务模型“editEquipment”（绑定“editEquipmentes”接口），执行接口绑定的“***HW***__editEquipment”脚本，将修改后的电梯记录更新到对象“***HW***__Equipment__CST”中。
删除	1. 通过自定义JS代码获取“表格”中，单击“删除”按钮的设备Id。 2. 将设备Id作为入参，运行服务模型“deleteEquipment”（绑定“deleteEquipment”接口），执行接口绑定的“***HW***__deleteEquipment”脚本，删除对象“***HW***__Equipment__CST”中的对应记录。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“Equipment”目录中，将鼠标放在“Page”上，单击界面上出现的  ，在弹出菜单中选择“标准页面”。

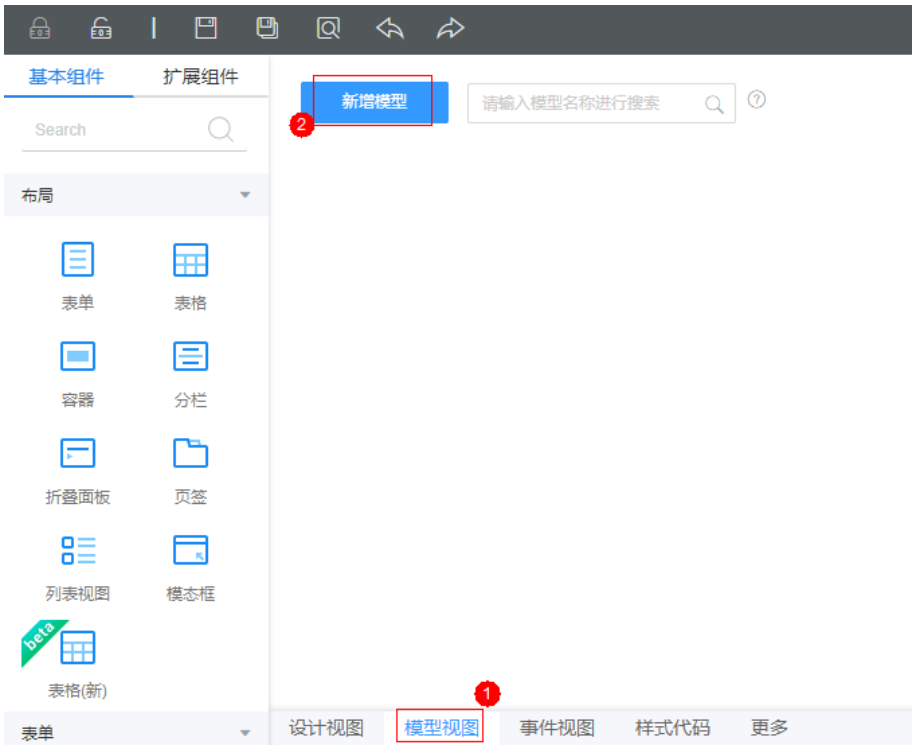
步骤3 在“标签”和“名称”文本框中输入“equipmentManage”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 定义自定义模型“equipmentInstance”。

1. 在“模型视图”中，单击“新增模型”。

图 2-125 新增模型



2. 添加自定义模型，模型名称“equipmentInstance”，单击“下一步”。
3. 单击“新增节点”，创建如表2-12所示的节点。字段名称要与设备对象为“***HW***__Equipment__CST”一致，然后单击“下一步”，再单击“确定”。

须知

加粗斜体请替换为实际的对象名、字段名。

表 2-12 新增节点

节点名称	数据类型
name	保持默认（Text）
HW__equipmentSN__CST	保持默认（Text）
HW__equipmentBrand__CST	保持默认（Text）
HW__equipmentModel__CST	保持默认（Text）
HW__fullAddress__CST	保持默认（Text）
HW__longitude__CST	保持默认（Text）
HW__latitude__CST	保持默认（Text）

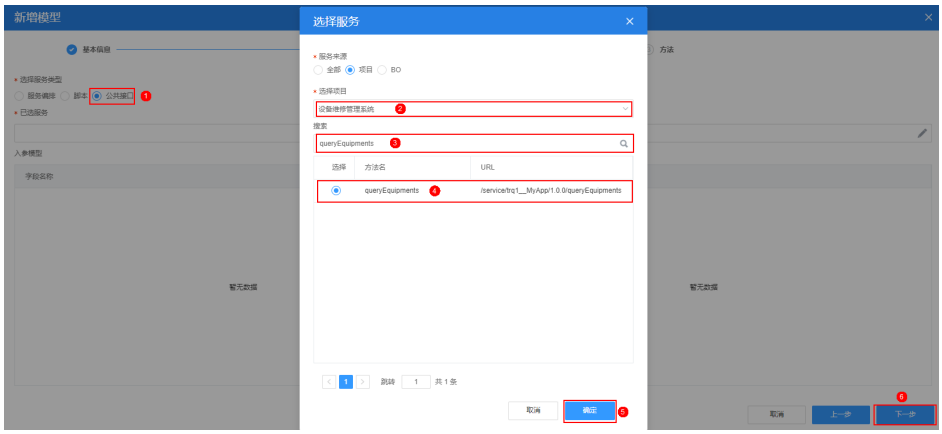
4. 单击页面上方的，保存设置。

步骤5 定义服务模型“queryEquipments”，与API（queryEquipments:1.0.0）关联。

1. 在“模型视图”中，单击“新增模型”，进入新增服务模型页面。
2. 模型名称设置为“queryEquipments”，来源选择“服务”，单击“下一步”。
3. 选择服务类型为“公共接口”，并“选择项目”为“设备维修管理系统”，指定模型与API“queryEquipments”关联，单击“确定”。

   关联API后，系统会自动显示API中脚本的输入、输出参数。
4. 在新增模型页面，单击“下一步”，方法保持不变，单击“确定”。
5. 单击页面上方的  ，保存设置。

图 2-126 定义服务模型



步骤6 定义服务模型“deleteEquipment”，与API（deleteEquipment:1.0.0）关联。

1. 在“模型视图”中，单击“新增模型”，进入新增服务模型页面。
2. 模型名称设置为“deleteEquipment”，来源选择“服务”，单击“下一步”。
3. 选择服务类型为“公共接口”，并“选择项目”为“设备维修管理系统”，指定模型与API“deleteEquipment”关联，单击“确定”。

   关联API后，系统会自动显示API中脚本的输入、输出参数。
4. 在新增模型页面，单击“下一步”，方法保持不变，单击“确定”。
5. 单击页面上方的  ，保存设置。

步骤7 切换到“设计视图”，从左侧基础组件区，拖拽一个“表格”到右侧“设计视图”中。

图 2-127 拖拽一个表格到页面

步骤8 为表格绑定模型，并设置表格查询结果区域。

1. 在“设计视图”中，选中“表格”，单击右侧“属性”页签，如图2-128所示。

图 2-128 为表格绑定对象模型

2. 在“选择模型”对话框中，选中“equipmentInstance”对象模型，单击“确定”。
   绑定对象模型后，系统自动将模型的所有字段添加为表格列，如图2-129所示。

图 2-129 绑定数据模型后的表格列

3. 选中上图中“已添加列”中的任意一项，上下拖动调整列的显示顺序。
4. 单击上图中每个字段后的，修改列标题等字段属性。
   以“设备编码”为例，其属性配置如图2-130所示。

图 2-130 设置“设备编码”列的属性

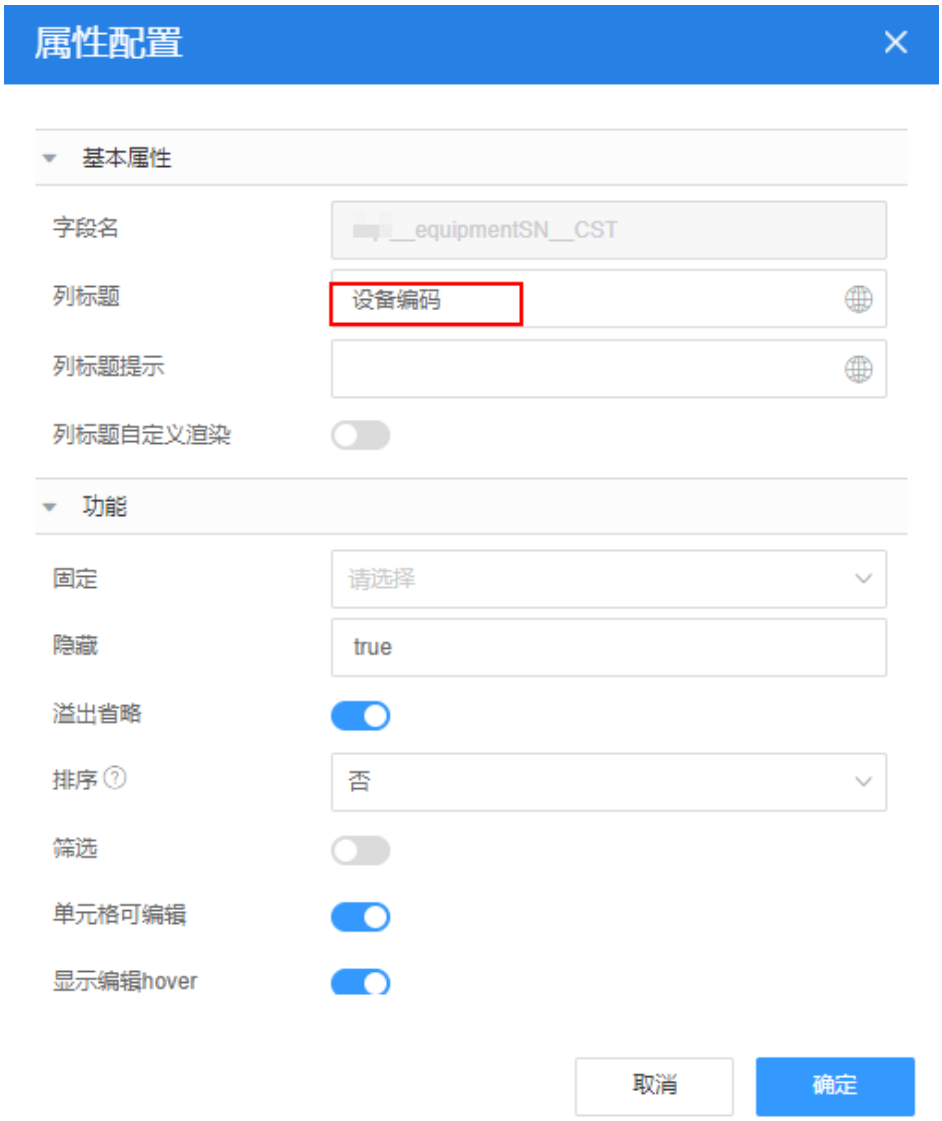


表 2-13 修改各字段的显示属性

调整后的顺序	字段名	列标题	可编辑
1	***HW***__equipmentSN__CST	设备编码	是
2	name	设备名称	是
3	***HW***__equipmentBrand__CST	品牌	是
4	***HW***__equipmentModel__CST	型号	是
5	***HW***__fullAddress__CST	地址	是
6	***HW***__longitude__CST	经度	是
7	***HW***__latitude__CST	纬度	是

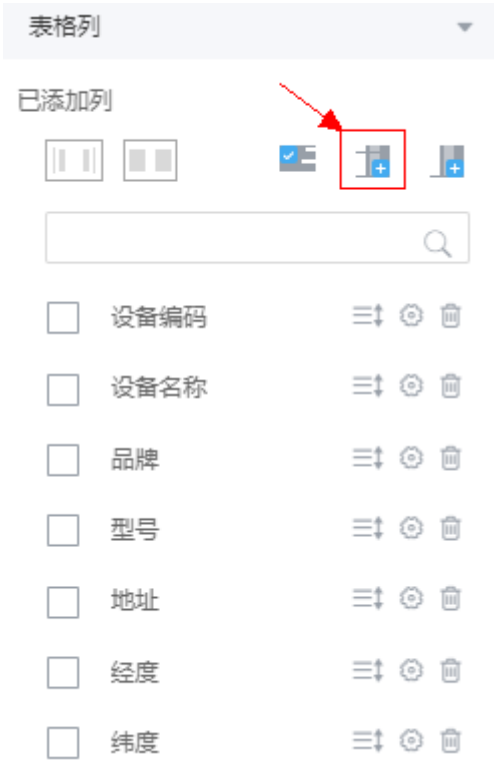
5. 单击页面上方的，保存设置。

步骤9 添加表格操作列，为表格添加编辑和删除两个操作按钮。

其中“编辑”是通过在按钮上添加自定义代码，将页面跳转到“编辑设备”页面。“删除”则是通过自定义代码调用“删除设备”脚本实现删除功能。

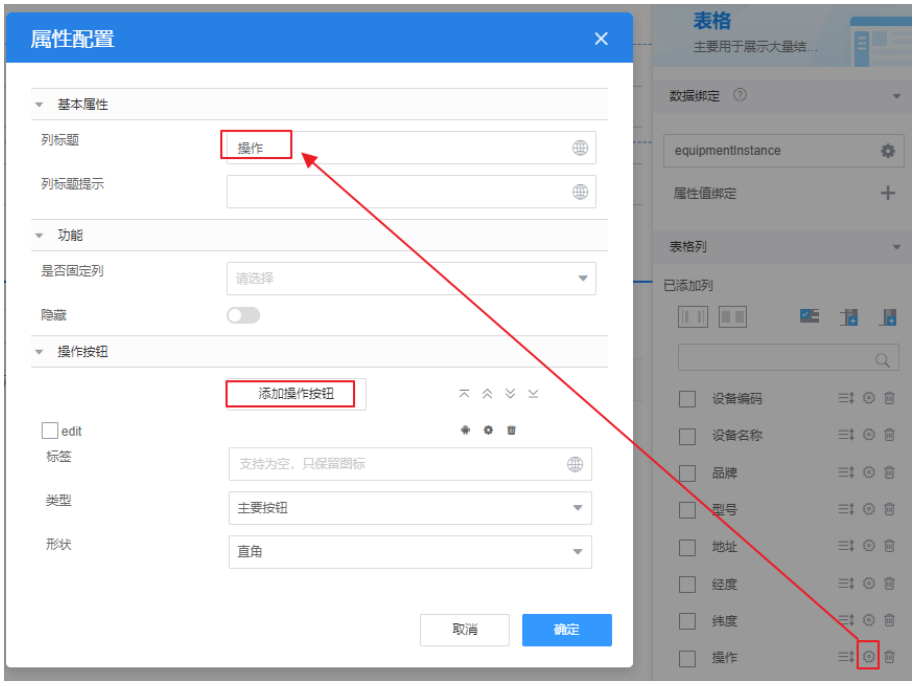
1. 在表格的“表格列”下，单击，添加表格操作列“Operation1”。

图 2-131 添加操作列



2. 单击新增的“Operation1”的，进入操作列属性配置。

图 2-132 设置操作列属性



3. 配置操作列属性。
   a. 将“列标题”设置为“操作”。
   b. 在操作按钮中，单击“添加操作按钮”2次，增加2个按钮。
   c. 设置第一个按钮“类型”为“主要按钮”，“图标”为“其他”页签中的“edit”，并单击，选择“动作列表”后的“+”，进入事件编辑，添加以下自定义代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名、页面名、组件ID。

```
//获取当前组件（即table）
let _component = context.$component.current;
//获取当前行row，取对象的id属性
let rowId = _component.$attrs.row.id;
//带着id跳转到设备详情页面
context.$page.load('/besBaas/page#/HW__editEquipment?id='+rowId);
```

图 2-133 设置操作列属性。

d. 向下拖动滚动条，设置第二个按钮“类型”为“主要按钮”，“图标”为“其他”页签中的“trash-a”，并单击，选择“动作列表”后的“+”，进入事件编辑，添加以下自定义代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名、页面名、组件ID。其中“table_*0*_condition”为当前表格组件的ID号，中间数字默认为“*0*”，如果有多次修改或创建该ID号会变化，选中表格后，在组件树上可以查看实际组件ID。

```
//获取当前组件（即table）
let _component = context.$component.current;
//获取当前行
let row = _component.$attrs.row;
// 表格组件
let _table =this.$component.table;
// 删除当前行数据
this.$dialog.confirm({
    title:'确认框',
    content:'确认是否删除？',
    okText:"确定",
    cancelText:"取消",
    onOk:()=>{
        $model.ref('deleteEquipment').setData({inputParam:{"id": row.id}});
        $model.ref('deleteEquipment').run().then(function(data){
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
```

```
let queryData ={
"start":(pageInfo.curPage -1)* pageInfo.pageSize,
"limit": pageInfo.pageSize
};
            $model.ref('queryEquipments').setData({inputParam: queryData});
            $model.ref('queryEquipments').run().then(function(data){
                $model.ref('equipmentInstance').setData(data.equipments);
                pageInfo.total = parseInt(data.total);
                $model.ref('table_0_condition').setData({"pageInfo": pageInfo});
}).catch(function(error){
                console.log('error is', error);
});
});
}
});
```

步骤10 设置工具栏区域。

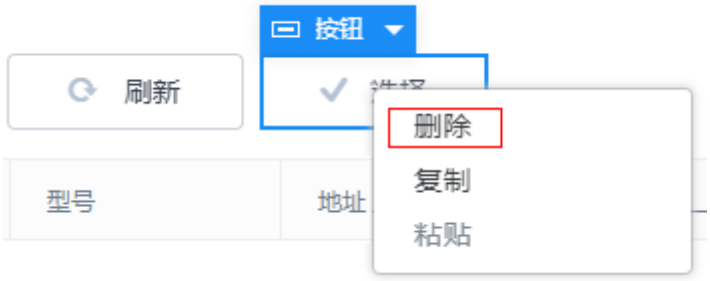
1. 在左侧“设计视图”中，选中“表格”，单击右侧“属性 > 表格区块”中“工具栏”后的“添加”按钮。

图 2-134 增加工具栏

2. 只保留“新增行”按钮，删除其他工具栏中多余的按钮（单击选中待删除按钮，鼠标右键单击，**请确认已选中按钮**，然后选择“删除”，如果误删其他组件，在页面上单击返回按钮即可）。

图 2-135 工具栏中需要删除的按钮

图 2-136 选中待删除按钮



3. 选中“新增行”按钮，在右侧属性面板中，修改“显示名称”为“新增设备”。在“事件”页签，再单击“新增行”的编辑按钮，在事件编排的“自定义动作”中，删除原来事件代码，输入以下事件代码，然后单击“创建”。

图 2-137 编辑事件代码


```
//跳转到编辑设备信息页面
context.$page.load('/besBaas/page#/HW__editEquipment');
```

4. 单击页面上方的，保存设置。

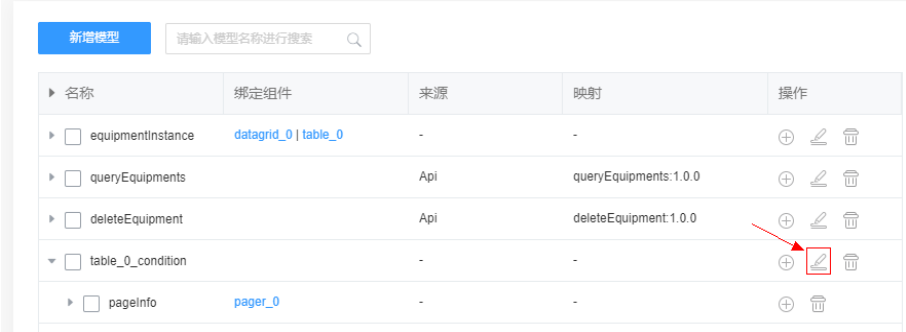
步骤11 创建查询条件区域。

1. 单击“模型视图”，切换到模型视图，在表格模型“table_*0*_condition”后，单击编辑图标。

   说明

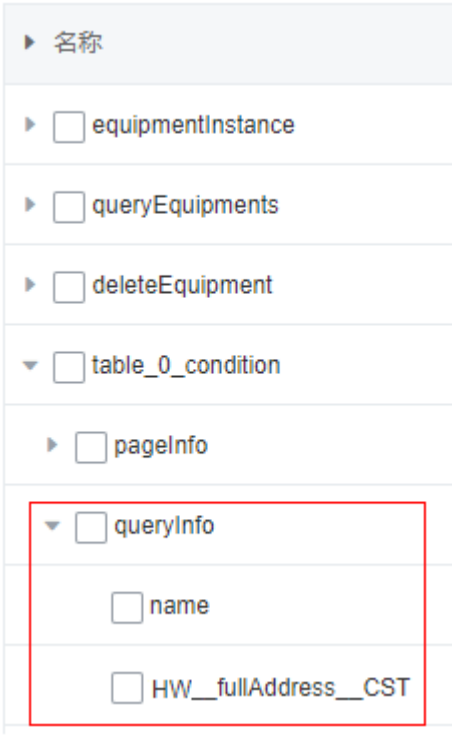
   表格组件拖入后，页面将自动生成一个表格模型。“table_*0*_condition”即是在前面拖入的表格对应的表格模型，模型名可能跟拖入顺序及个数有关，一般情况下模型名称为“table_*0*_condition”，如果名称不一致，请根据实际情况修改。

图 2-138 编辑表格模型



2. 在表格模型“table_***0***_condition”中，单击“新增节点”，添加一个“queryInfo”节点，“字段类型”为“Any”，然后在“queryInfo”下，再单击“新增节点”添加2个子节点“name”、“***HW***__fullAddress__CST”。

图 2-139 创建完成后的模型节点



3. 单击“下一步”，单击“确定”，完成模型修改。
4. 单击“设计视图”，切换到设计视图，从左侧组件列表中，拖一个“表单”组件到“表格容器”的最上部（表格容器内部最上部），在“元数据表单配置向导”弹窗中，单击“取消”。

图 2-140 向表格容器拖入表单

5. 选中表单，在属性中“数据绑定”下，单击，为表单绑定“table_***0***_condition”中的“queryInfo”节点，并在提示弹窗中，单击“绑定并生成表单”，单击“确定”。

图 2-141 为表单绑定数据对象字段

图 2-142 选择绑定并生成表单

6. 在表单中，选择“name”输入框，修改“标签”为“设备名称”、选择“***HW***__fullAddress__CST”输入框，修改“标签”为“地址”。

7. 选中“保存”按钮，修改“显示名称”为“查询”。在“事件”页签，单击，删除“提交表单”事件，然后单击“点击”后的“+”，进入事件编排，删除原有自定义代码，输入以下事件代码，单击“创建”。

图 2-143 修改按钮事件代码

须知

脚本中**红色内容**，请替换为实际的对象名、字段名、页面名、组件ID。其中“table_*0*_condition”为当前表格组件的ID号，中间数字默认为“*0*”，如果有多次修改或创建该ID号会变化，选中表格后，在组件树上可以查看实际组件ID。

```
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
let queryInfo = $model.ref('table_0_condition').getData().queryInfo;
if (!queryInfo) {
    queryInfo = {};
}
let queryData = {
    "fullAddress": queryInfo.HW__fullAddress__CST,
    "name": queryInfo.name,
    "start": 0,
    "limit": pageInfo.pageSize
};
$model.ref('queryEquipments').setData({inputParam: queryData});
$model.ref('queryEquipments').run().then(function (data) {
    $model.ref('equipmentInstance').setData(data.equipments);
    pageInfo.total = parseInt(data.total);
    $model.ref('table_0_condition').setData({"pageInfo": pageInfo});
}).catch(function (error) {
    console.log('error is', error);
});
```

查询条件区域创建后，如图2-144所示。

图 2-144 页面预览

8. 选中“重置”按钮，在“事件”页签，单击，删除“重置表单”事件，然后单击“点击”后的“+”，进入事件编排，删除原有自定义代码，输入以下事件代码，单击“创建”。

图 2-145 修改重置按钮事件代码

须知

脚本中**红色内容**，请替换为实际的对象名、字段名、页面名、组件ID。其中“table_*0*_condition”为当前表格组件的ID号，中间数字默认为“*0*”，如果有多次修改或创建该ID号会变化，选中表格后，在组件树上可以查看实际组件ID。

```
var table_0_condition = $model.ref("table_0_condition").getData();
table_0_condition.queryInfo = {};
$model.ref("table_0_condition").setData(table_0_condition);
```

9. 单击页面上方的，保存设置。

步骤12 添加页面事件代码。

1. 在“设计视图”中，选中最外层的“页面”，可以直接单击组件导航，快速定位。
2. 在右侧“事件”页签中，单击“加载”后的“+”，为页面添加事件代码。

3. 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名、页面名、组件ID。其中“table_*0*_condition”为当前表格组件的ID号，中间数字默认为“*0*”，如果有多次修改或创建该ID号会变化，选中表格后，在组件树上可以查看实际组件ID。

```
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
let queryData ={
"start":(pageInfo.curPage -1)* pageInfo.pageSize,
"limit": pageInfo.pageSize
};
$model.ref('queryEquipments').setData({ inputParam: queryData });
$model.ref('queryEquipments').run().then(function(data){
    $model.ref('equipmentInstance').setData(data.equipments);
    pageInfo.total = parseInt(data.total);
    $model.ref('table_0_condition').setData({"pageInfo": pageInfo });
}).catch(function(error){
    console.log('error is', error);
});
```

4. 单击“创建”，关闭事件编排器，返回到页面。
5. 单击页面上方的  ，保存页面。

----结束

验证

单击界面上方的  ，进入预览页面，查看页面的展示效果，并验证以下功能：

步骤1 在界面上单击“新增设备”，查看是否跳转到“编辑设备”页面，如未跳转，请检查事件代码中页面名称前缀是否是实际空间名。

步骤2 检查“编辑设备”页面的“设备品牌”下拉框，“省市区”级联框的选项是否正确。

选项正确，则说明自定义模型、页面组件与模型绑定关系、以及页面on-load事件是正确的。

步骤3 检查“编辑设备”页面（***HW***__editEquipment）的设备录入功能是否正确。

1. 填写“设备名称”、“设备编码”等信息， 单击“保存”。
2. 检查“设备管理”页面（***HW***__equipmentManage）是否包含新插入的数据。

   如果查询结果页面显示新增数据，则说明新增功能以及页面的事件代码正确。

   如果操作不成功，可以在页面事件脚本中增加debugger，使用Chrome开发者工具来调试定位问题。

步骤4 检查“编辑设备”页面（***HW***__editEquipment）的设备修改功能是否正确。

1. 单击一条记录后面的编辑按钮，查看是否跳转到“编辑设备”页面。
2. 修改“设备名称”、“设备编码”等信息， 单击“保存”。
3. 检查“设备管理”页面（***HW***__equipmentManage）当前设备记录是否更新。

   如果查询结果页面显示更新数据，则说明修改功能以及页面的事件代码正确。

步骤5 检查“设备管理”页面（***HW***__equipmentManage）的设备删除、查询、重置功能是否正确。

1. 选中空白行或其他行，单击“删除”，验证删除功能是否正常，如果不能删除，请检查操作列“删除”按钮事件代码。
2. 在查询区域分别输入“设备名称”、“地址”，单击“搜索”或“重置”验证查询电梯设备信息功能，如果不能查询或重置，请检查操作列按钮的事件代码。

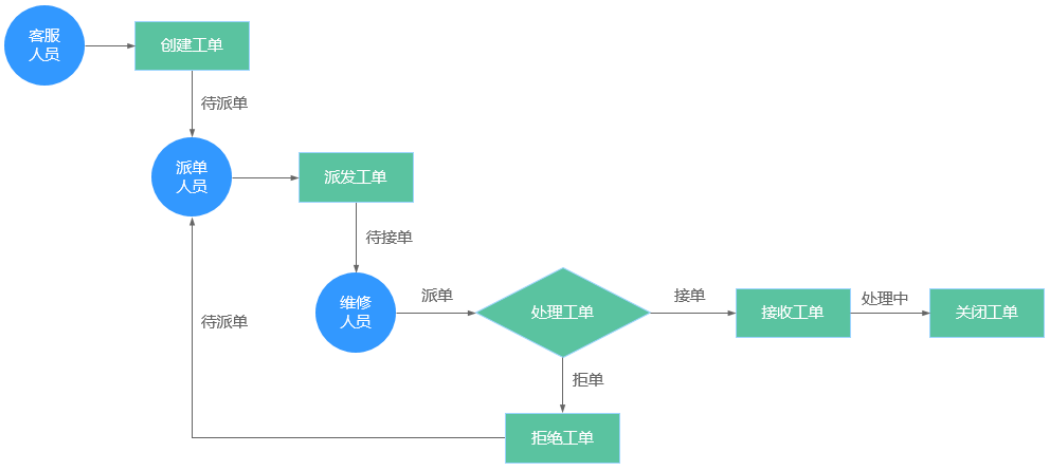
----结束

2.7 工单管理开发

2.7.1 开发前必读

工单管理功能是通过创建脚本，实现工单创建、工单派发以及工单处理等功能。通过创建的BPM，实现工单在客服人员、派单人员、维修人员之间流转过程，如图2-146。

图 2-146 工单流转大致流程



1. 电梯客服人员受理用户投诉，并创建维修工单。
2. 派单人员收到客服人员的维修工单后，派发给维修工程师。
3. 维修工程师进行现场修理，并在处理完成后关闭维修单。

2.7.2 定义数据对象“工单对象”

对于工单对象功能，需要先创建一个工单对象WorkOrder，保存工单、型号、资产编号等信息，如表2-14所示。

表 2-14 工单对象 WorkOrder 信息

字段标签	字段名称	字段描述	数据类型	建议长度	备注
instanceId	instanceId	BPM实例Id	文本	255	-
createdBy	createdBy	工单创建人	文本	255	-
equipmentId	equipmentId	设备ID	文本	255	-

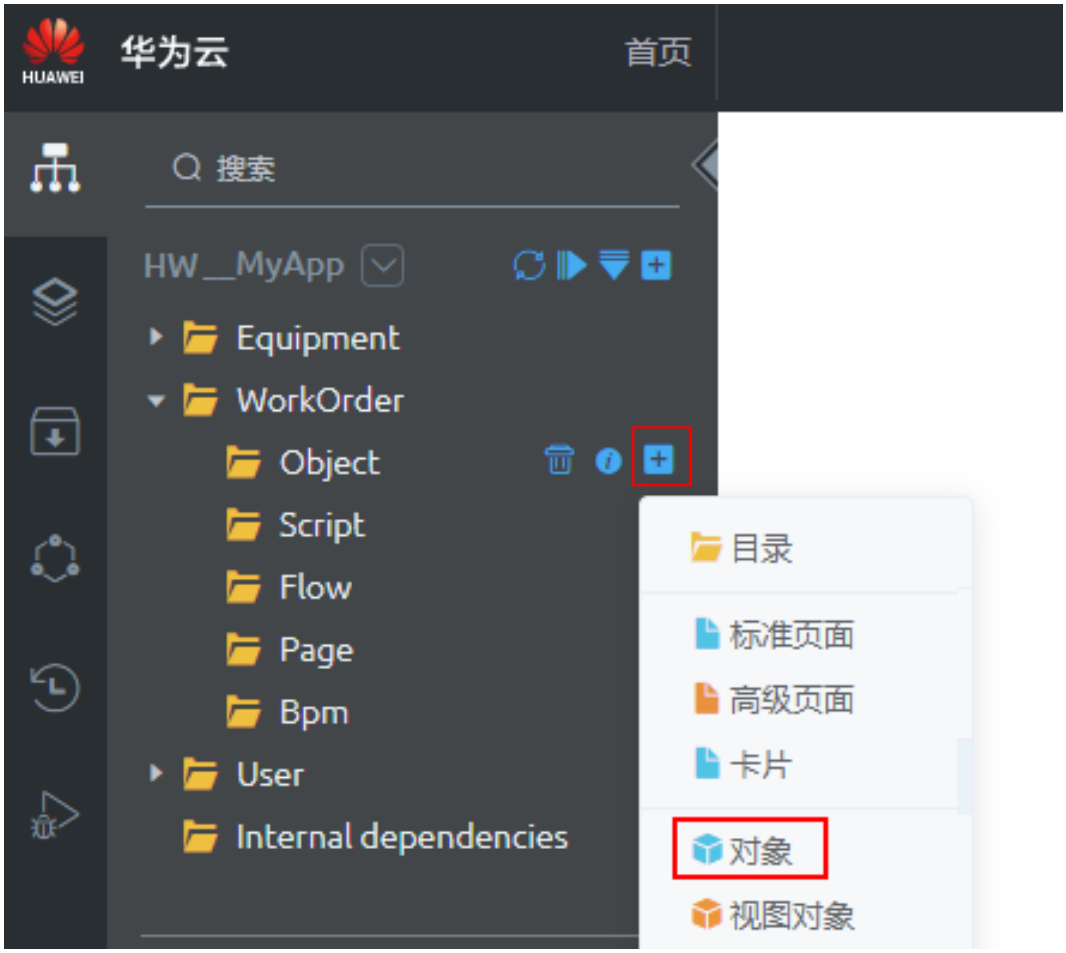
字段标签	字段名称	字段描述	数据类型	建议长度	备注
type	type	工单类型	文本	255	-
description	description	备注	文本	255	-
source	source	来源	文本	255	-
faultPhenomenon	faultPhenomenon	故障现象说明	文本	255	-
arriveTime	arriveTime	客户要求工程师到达现场时间	日期/时间	-	-
recoveryTime	recoveryTime	客户要求故障恢复时间	日期/时间	-	-
customer	customer	客户接口人	文本	255	-
assignedFme	assignedFme	当前处理人	文本	255	-
title	title	现场服务单标题	文本	255	-
status	status	服务单状态	文本	255	-
priority	priority	服务单优先级	文本	255	-
workOrderId	workOrderId	服务单ID	文本	255	区分大小写、是否必选（否），是否唯一（是）、是否可搜索（是）

操作步骤

步骤1 在经典版开发环境“首页 > 我的应用”中，单击“设备维修管理系统”，进入之前创建的应用。

步骤2 在“WorkOrder”中，将鼠标放在“Object”目录上，单击界面上出现的“+”，在弹出菜单中选择“对象”。

图 2-147 创建对象



步骤3 选中“创建新对象”，在“标签”和“名称”文本框中输入“WorkOrder”，单击“添加”。

系统实际创建的对象名称为“***HW***__WorkOrder__CST”，“***HW***__”前缀由租户命名空间决定，“__CST”后缀代表是自定义对象。

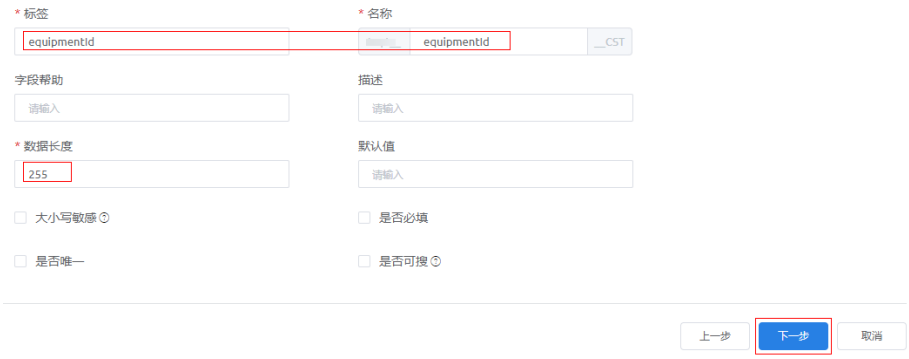
步骤4 按系统向导完成第1个字段“设备ID”（equipmentId）的定义。

1. 在“自定义字段”页签，单击“新建”。
2. 选中字段类型，设置字段类型为“文本”，单击“下一步”。
3. 输入字段详细信息，如图2-148所示，设置字段标签、名称为“equipmentId”、字段长度255，单击“下一步”。

须知

平台会根据字段标签自动生成字段名称，但请参照表2-14，修改字段名。对于utf-8编码，一个汉字占用三个字节。

图 2-148 定义“设备 ID”详细信息

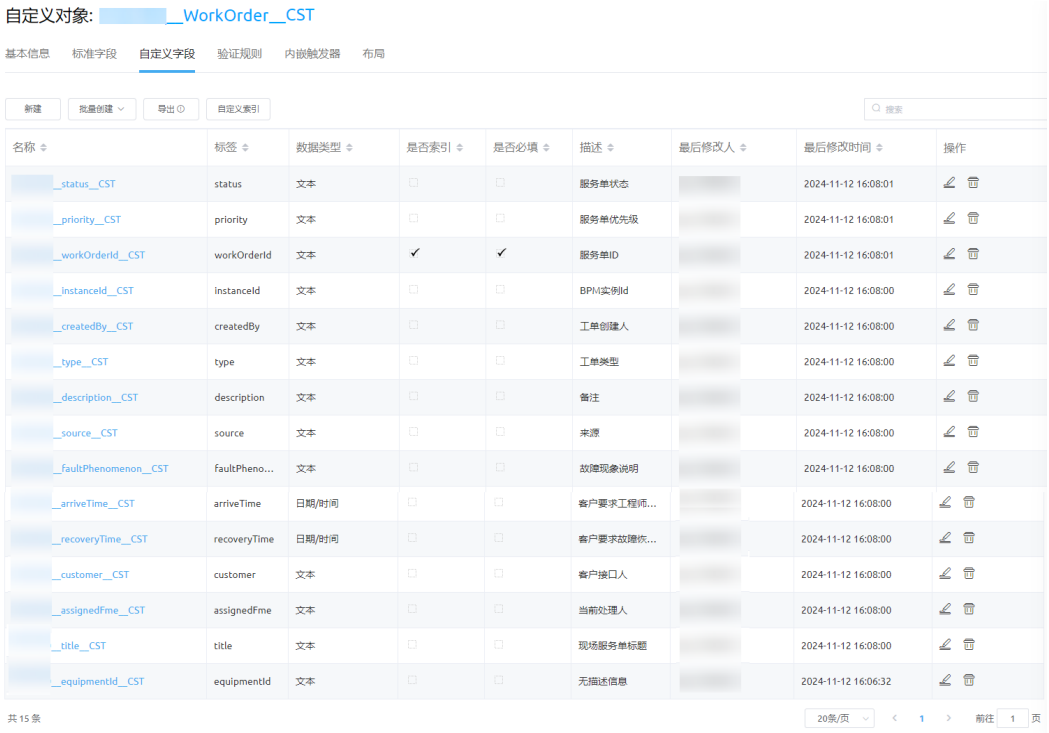


系统实际创建的字段名称为“***HW***__equipmentId__CST”，“***HW***__”前缀由租户命名空间namespace决定，“__CST”后缀代表是自定义字段。

4. 配置字段级安全，选中“读取”和“编辑”复选框，为所有预置profile配置能编辑和读取本字段的权限，单击“下一步”。
5. 将字段添加到页面布局，选中“添加本字段到该页面布局”，单击“保存”。

步骤5 按照上述操作，创建**表2-14**中其他字段。

图 2-149 自定义对象字段



----结束

2.7.3 开发“客服人员创建工单”功能

2.7.3.1 本节导读

“客服人员创建工单”包含客服人员生成工单和查询工单两个功能，这两个功能分别通过两个标准页面实现：

- 生成工单页面：客服人员输入工单信息，生成工单页面。
- 工单列表（客服人员）页面：客服人员在工单列表页面查询工单信息，查看所有工单的列表页面。

学习地图

如图2-150所示，通过本章的学习和实践，您将进一步了解“标准页面”的能力，包括：

- 拼装复杂页面
- 了解折叠面板
- 学习表格：自定义某查询结果字段的显示内容
- 动态加载页面内容
- 了解在事件中调用脚本的API

图 2-150 学习地图

2.7.3.2 开发“生成工单”功能

2.7.3.2.1 创建“生成工单”脚本及公共接口

“生成工单”功能对应电梯管理流程的第一个节点，客服人员在录入完成维修信息、提交维修单时将会调用“生成工单”脚本及公共接口。

创建“生成工单”脚本

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“createWorkOrder”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于创建工单
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import * as date from 'date';
//定义入参结构，入参包含1个参数：workOrder对象，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'Object', required: true, label: 'object' })
    workOrderData: object;
}
//定义出参结构，出参包含1个参数，workOrder的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
}
//使用数据对象HW__WorkOrder__CST
@useObject(['HW__WorkOrder__CST'])
@action.object({ type: "method" })
export class CreateWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public createWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let workOrderData = input.workOrderData;    //将入参赋值给workOrderData变量，方便后面使用
            let s = db.object('HW__WorkOrder__CST');    //获取HW__WorkOrder__CST这个Object的操作实例
            let userName = context.getUserName();
            workOrderData['HW__status__CST'] = '待派单';
            workOrderData['HW__assignedFme__CST'] = '派单员';
            workOrderData['HW__createdBy__CST'] = userName;
            let id = s.insert(workOrderData);
            if (id) {
                out.id = id;
            } else {
                error.name = "WOERROR";
                error.message = "无法创建工单！";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的  ，执行脚本。
2. 在界面底部输入测试数据，单击测试窗口右上角  执行图标。
   测试报文样例如下:

   ```
   {
     "workOrderData":{
   ```



```
    "createdDate":"2020-09-30 12:00:00",
    "HW__recoveryTime__CST":"2020-09-30 12:00:00",
    "HW__arriveTime__CST":"2020-09-30 12:00:00",
    "HW__title__CST":"电梯无法关门",
    "HW__priority__CST":"高",
    "HW__workOrderId__CST":"WD0000123456",
    "HW__instanceId__CST":""
  }
}
```

执行成功，会在“输出”页签返回查询结果。

图 2-151 输出工单 ID 号

步骤7 测试成功，单击编辑器上方的，启用脚本。

----结束

创建“生成工单”公共接口

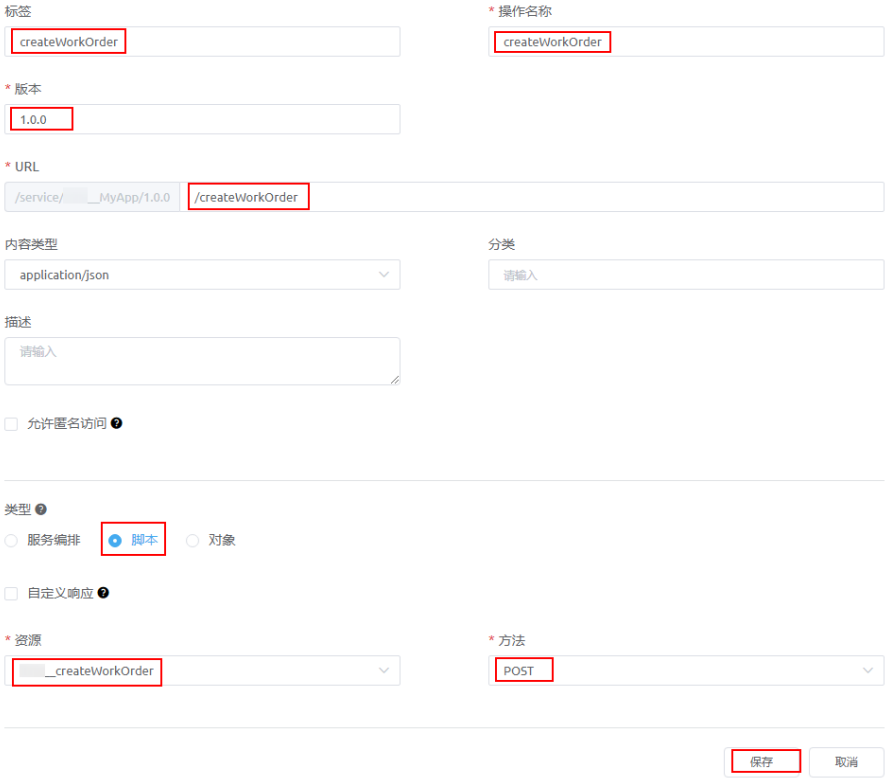
步骤1 在APP视图下，单击“服务”，进入公共接口创建页面。

图 2-152 创建公共接口入口

步骤2 单击“新建”，创建“生成工单”脚本“***HW***__createWorkOrder”的公共接口。

设置接口参数信息：操作名称为“createWorkOrder”，版本为“1.0.0”，URL为“/createWorkOrder”，“类型”选择“脚本”，“资源”为“***HW***__createWorkOrder”，方法为“POST”，然后单击“保存”。

图 2-153 新建 createWorkOrder 接口



----结束

2.7.3.2.2 组装“生成工单”页面

“生成工单”对应设备管理流程的第一个节点，客服人员使用这个页面完成维修信息录入，生成维修工单，并提交给派单人员处理。该页面使用标准页面组装实现。

页面组件分析

“生成工单”页面的组装过程如下：

1. 在页面中，拖入一个**容器**组件。
2. 在**容器**组件中，拖入一个**表单**组件和**容器**组件。
   a. 将**表单**组件分割为**3个区域**：“基本信息”、“设备信息”、“其他信息”。
   b. 将2级**容器**作为“按钮”区域。

组装完成后，整个页面有4个区域，如图2-154所示。

图 2-154 生成工单页面

其中，“基本信息”、“设备信息”是通过数据绑定页面模型直接创建的表单，“其他信息”及“按钮”区域是手动拖入的组件，然后再进行数据绑定，页面中详细组件分布如图2-155所示。

图 2-155 页面组件分析

页面模型分析

页面模型负责与页面组件交互，获取生成工单需要的数据，页面模型需要先定义，然后再与页面组件进行绑定。

对于设备维修工单，需要录入的信息包括：工单基本信息、设备信息以及工单扩展信息。

打开创单页面，有如下处理逻辑：

1. 在“基本信息”区域指定“设备名称”时，在“设备详情”区域自动显示设备信息。该能力需要开发（equipmentSelectListQuery）脚本实现。
2. 填写完工单，单击“提交”按钮后，会创建一个工单实例。该能力通过调用BPM实现。

结合填单页面需求，以及前端组件对应关系，需要创建如下模型：

表 2-15 模型分析

模型名称	作用	来源	详细定义
basicInfo	保存工单的基本信息，将与“基础信息”、“其他信息”区域中各个组件绑定	自定义模型	新增的字段名称请与下面字段名称保持一致，后续将绑定页面组件。包含的节点如下，这些节点与前台页面上的工单基本信息一一对应，加粗斜体部分要替换为自己账号对应的命名空间： ● ***HW***__equipmentId__CST：设备ID，字段类型**SingleSelect**。 ● ***HW***__title__CST：工单标题，字段类型Text。 ● createdDate：创建时间，字段类型**DateTime**。 ● ***HW***__type__CST：工单类型，字段类型**SingleSelect**。 ● ***HW***__priority__CST：工单优先级，字段类型**SingleSelect**。 ● ***HW***__description__CST：工单备注说明，字段类型**TextArea**。 ● ***HW***__faultPhenomenon__CST：故障现象说明，字段类型Text。 ● ***HW***__customer__CST：客户接口人，字段类型Text。 ● ***HW***__recoveryTime__CST：客户要求故障恢复时间，字段类型**DateTime**。 ● ***HW***__source__CST：工单来源，字段类型Text。 ● ***HW***__arriveTime__CST：客户要求工程师到达现场时间，字段类型**DateTime**。

模型名称	作用	来源	详细定义
equipmentInfo	获取设备详细信息，将与“设备信息”区域组件绑定	自定义模型	将绑定页面组件。包含的计算节点如下，这些节点与前台页面上的设备信息一一对应，加粗斜体部分要替换为自己账号对应的命名空间： • name：设备名称，字段类型Text。 • ***HW***__equipmentSN__CST：设备编码，字段类型Text。 • ***HW***__equipmentBrand__CST：设备品牌，字段类型Text。 • ***HW***__equipmentModel__CST：设备型号，字段类型Text。 • ***HW***__fullAddress__CST：设备完整地址，字段类型Text。 **须知** 这些节点名称需要与设备对象的字段名保持一致。
equipmentList	查询系统中的设备，设备名称下拉框的属性值绑定模型	自定义模型	与“设备名称”下拉框绑定，将从下拉框的值赋给当前下拉框值。
equipmentOptions	查询系统中的设备，设备下拉框的可选项	服务模型	与公共接口equipmentSelectListQuery关联。
setInstanceId	查询系统中的设备，设备下拉框的可选项	服务模型	与公共接口createWorkOrder关联。

操作步骤

步骤1 进入**创建“设备维修管理系统”应用**中创建的应用。

步骤2 在“WorkOrder”目录中，鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“createWorkOrder”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击进行锁定。

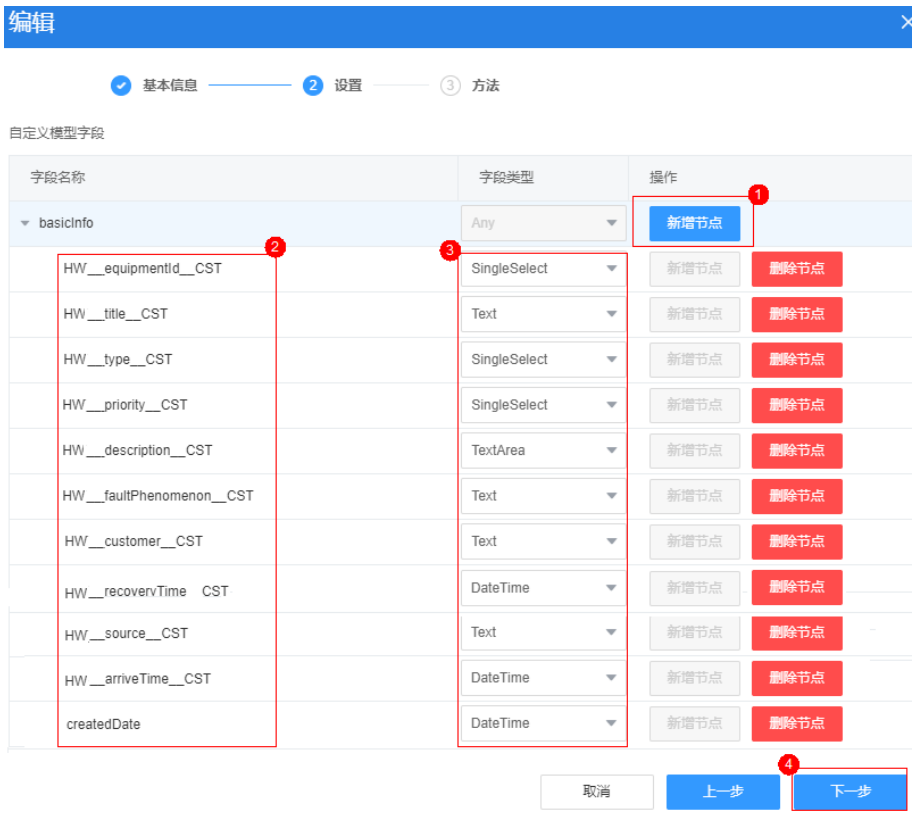
步骤4 在标准页面的“设计视图”下，单击页面底部的“模型视图”页签，切换到“模型视图”。

图 2-156 切换模型视图

步骤5 定义模型“basicInfo”。

1. 在模型视图页面，单击“新增模型”。
2. 添加自定义模型，模型名称“basicInfo”，单击“下一步”。
3. 单击“新增节点”，新增设备ID字段“***HW***__equipmentId__CST”，并设置字段类型“SingleSelect”，加粗斜体内容请替换为实际命名空间前缀。
4. 依次增加表2-16中列出的字段名称，单击“下一步”，再单击“确定”。

图 2-157 新增节点字段



须知

加粗斜体内容请替换为实际命名空间前缀。

表 2-16 新增 basicInfo 模型的节点字段

字段名称	字段类型	字段描述
HW__equipmentId__CST（上一步已添加）	**SingleSelect**	设备ID
HW__title__CST	Text	工单标题
createdDate	**DateTime**	创建时间
HW__type__CST	**SingleSelect**	工单类型

字段名称	字段类型	字段描述
HW__priority__CST	**SingleSelect**	工单优先级
HW__description__CST	**TextArea**	工单备注说明
HW__faultPhenomenon__CST	Text	故障现象说明
HW__customer__CST	Text	客户接口人
HW__recoveryTime__CST	**DateTime**	客户要求故障恢复时间
HW__source__CST	Text	工单来源
HW__arriveTime__CST	**DateTime**	客户要求工程师到达现场时间

5. 单击页面上方的  ，保存设置。

步骤6 定义模型“equipmentList”。

1. 在模型视图页面，单击“新增模型”。
2. 添加自定义模型，模型名称“equipmentList”，单击“下一步”。
3. 单击“下一步”，再单击“确定”。
4. 单击页面上方的  ，保存设置。

步骤7 定义模型“equipmentInfo”。

1. 在模型视图页面，单击“新增模型”。
2. 添加自定义模型，模型名称“equipmentInfo”，单击“下一步”。
3. 单击“新增节点”，依次新增表2-17中列出的字段名称，单击“下一步”，再单击“确定”。

须知

加粗斜体内容请替换为实际命名空间前缀。

表 2-17 新增 equipmentInfo 模型的节点字段

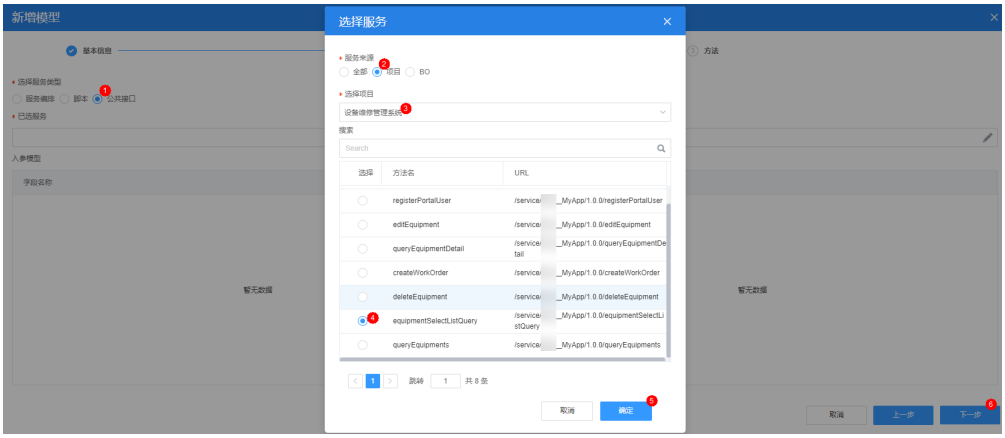
字段名称	字段类型	字段描述
name	Text	设备名称
HW__equipmentSN__CST	Text	设备编码
HW__equipmentBrand__CST	Text	设备品牌
HW__equipmentModel__CST	Text	设备型号
HW__fullAddress__CST	Text	设备完整地址

4. 单击页面上方的，保存设置。

步骤8 定义模型“equipmentOptions”。

1. 在模型视图页面，单击“新增模型”。
2. 添加服务模型，模型名称“equipmentOptions”，单击“下一步”。
3. 设置服务类型为“公共接口”，“项目”选择“设备维修管理系统”，公共接口为“equipmentSelectListQuery”，即**创建“查询设备列表”脚本**中定义的脚本“equipmentSelectListQuery”对应的公共接口，单击“下一步”，再单击“确定”。

图 2-158 选择服务



4. 单击页面上方的，保存设置。

步骤9 定义模型“setInstanceId”。

1. 在模型视图页面，单击“新增模型”。
2. 添加服务模型，模型名称“setInstanceId”，单击“下一步”。
3. 设置服务类型为“公共接口”，“项目”选择“设备维修管理系统”，公共接口为“createWorkOrder”，即“生成工单”脚本“createWorkOrder”对应的公共接口，单击“下一步”，再单击“确定”。
4. 单击页面上方的，保存设置。

步骤10 拖拽页面框架组件（“基本信息”、“设备信息”、“其他信息”以及“按钮”区域）。

1. 单击页面底部的“设计视图”页签，切换到“设计视图”。
2. 从基本组件列表区拖拽“容器”到右侧“页面内容”中，然后再拖拽一个“表单”到“容器”中，并在“元数据表单配置向导”弹窗中，单击“取消”。
3. 拖拽一个“容器”到“表单”下方（**表单之外**），作为按钮区域的容器。

图 2-159 拖拽按钮区域容器

4. 选择“表单”，拖一个“折叠面板”组件到“表单”中，并在页面右下角的组件树中，删除2个“折叠页”（在组件树中，选中后单击鼠标右键，再单击“删除”即可），删除后折叠面板包含1个折叠页。

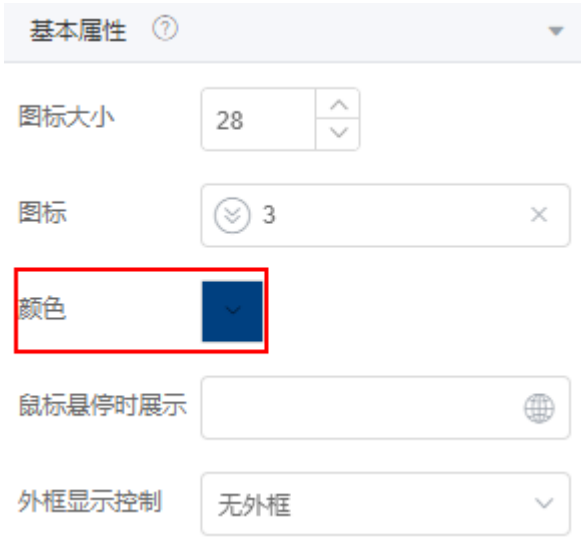
图 2-160 组件层级关系及删除多余折叠页

5. 在“折叠面板”中，选中“折叠页签标题”，然后在右侧“属性”页签中进行以下设置：
   - 在“样式”下的“背景”，将背景颜色修改为浅蓝色（#9BCCEF）。

图 2-161 设置背景为浅蓝色

- 选中图标，设置图标颜色为深蓝色（#004080）。

图 2-162 设置图标颜色为深蓝色



- 选中图标后的“标签”，设置“文本内容”为“基本信息”。

图 2-163 设置标签内容为基本信息

您还可以根据需要，修改边距等其他样式，使其更美观。

6. 展开组件树，将鼠标放在折叠面板上，单击右键选择“复制”，再单击右键选择2次“粘贴”，在折叠面板之后，添加2个同样的折叠面板。

图 2-164 粘贴后的折叠面板

7. 分别设置第二、第三个折叠面板的“折叠页签标题”为“设备信息”、“其他信息”。

图 2-165 修改折叠页标题

8. 单击界面上方的，保存页面。

步骤11 组装“基本信息”区域。

1. 在“基本信息”所在的折叠面板下，从左侧组件列表中，拖一个“表单”到“折叠页签内容”中，然后在“元数据表单配置向导”弹窗中，单击“取消”。

图 2-166 拖入表单

2. 选中“表单”，在右侧属性中，单击，在模型选择弹窗中选中“basicInfo”，单击“确定”，并在提示弹窗中，单击“绑定并生成表单”。

图 2-167 表单数据绑定

3. 在生成的表单中，删除如表2-18中的多余字段所在组件以及按钮。

表 2-18 需要删除的组件名及组件

字段名	组件
HW__faultPhenomenon__CST	输入框
HW__customer__CST	输入框
HW__recoveryTime__CST	日期选择框
HW__arriveTime__CST	日期选择框
保存、重置	按钮

4. 分别选择表单中的2个“分栏”，在“行布局”中单击  ，修改分栏为4列，并按图2-169所示调整字段组件的位置（选中组件，用鼠标即可拖拽组件到目的栏），调整完成后，删除多余空白分栏。

图 2-168 调整分栏为 4 列

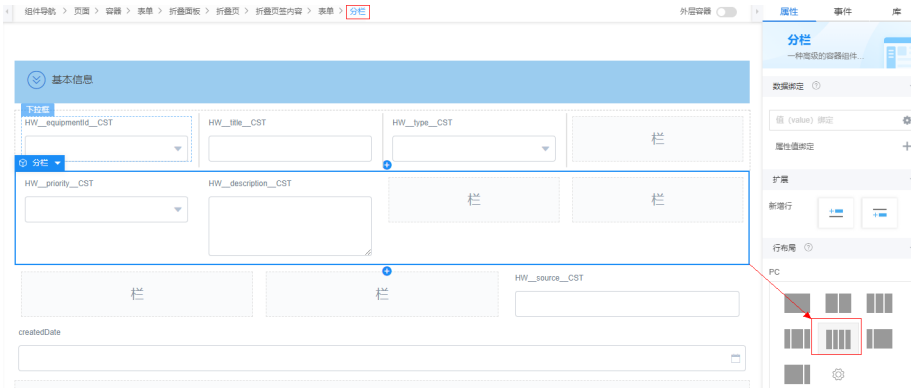
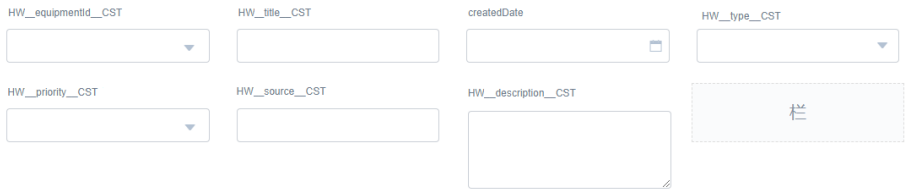


图 2-169 调整后表单



5. 选择表单中的各个组件，依次在右侧属性面板中修改各个组件标签，详细标签名如表2-19所示。

表 2-19 修改表单组件名及组件对应字段

标签名	字段名	组件类型
设备名称	***HW***__equipmentId__CST	下拉框
工单标题	***HW***__title__CST	输入框
创建时间	createdDate	日期选择框
工单类型	***HW***__type__CST	下拉框
优先级	***HW***__priority__CST	下拉框
工单来源	***HW***__source__CST	输入框
备注	***HW***__description__CST	多行输入框

6. 选中“设备名称”下拉框，在右侧属性页签，单击“属性值绑定”的“+”，“属性”设置为“选项”，然后再单击“模型字段”的设置图标，在弹窗中勾选“equipmentList”模型，对“设备名称”下拉框进行属性绑定。

图 2-170 添加属性值绑定

7. 选中“工单类型”下拉框，在右侧属性页签中，设置“维修”和“保养”2个选项。

图 2-171 添加工单类型

8. 选中“优先级”下拉框，在右侧属性页签中，设置“高”、“中”和“低”3个选项。

图 2-172 添加优先级

9. 单击界面上方的，保存页面。

步骤12 组装“设备信息”区。

1. 在“设备信息”所在的“折叠面板”中，拖入一个“表单”组件到“折叠页签内容”中，并在“元数据表单配置向导”弹窗中，单击“取消”。
2. 选择“表单”，在右侧属性中，单击，在弹窗中选中“equipmentInfo”，单击“确定”，并在提示中单击“绑定并生成表单”。

图 2-173 表单数据绑定

3. 删除“name”输入框、“保存”、“取消”按钮以及按钮所在“容器”。
4. 选择“分栏”，在“行布局”中单击，修改分栏组件为4列，并删除空白分栏，调整后如图2-174所示。

图 2-174 调整后表单

5. 修改各个输入框组件的标签名，并“公共”属性下，设置“只读”。

表 2-20 组件名及对应字段

标签名	字段名	其他属性
设备编码	***HW***__equipmentSN__CST	只读
设备品牌	***HW***__equipmentBrand__CST	只读
设备型号	***HW***__equipmentModel__CST	只读
地址	***HW***__fullAddress__CST	只读

6. 单击页面上方的  ，保存页面。

步骤13 组装“其他信息”区域，组装完成后如图2-176所示。

图 2-175 其他信息区域组件结构

图 2-176 其他信息区域

1. 在“其他信息”所在的“折叠面板”中，拖1个“表单”组件到“折叠页签内容”中，然后单击“取消”元数据表单配置向导弹窗。
2. 设置表单样式：选中“表单”组件，在“样式”下“高级设置”中，设置表单样式“:root{margin-left:50px;}”。

3. 在“表单”中，拖入1个“折叠面板”，并删除1个折叠页，保留2个折叠页。
4. 修改第1个折叠页“折叠页签标题”的“标签”下的“文本内容”修改为“故障记录”。
5. 在“故障记录”的“折叠页签内容”下，拖入1个分栏组件，设置为3列，并向第1列拖入“输入框”，并修改输入框的标签为“故障现象说明”。
6. 修改第2个折叠页“折叠页签标题”的“标签”下的“文本内容”修改为“时间记录”。
7. 在“时间记录”的“折叠页签内容”下，拖入1个分栏组件，并设置为3列，并分别向第1列和第2列拖入1个“日期选择框”，修改“标签”为“客户要求工程师到达现场时间”、“客户要求故障恢复时间”，然后修改“类型”为“日期时间”并设置“日期格式”为“yyyy-MM-dd HH:mm:ss”，向第3列拖入一个输入框，设置“标签”为“客户接口人”。

图 2-177 设置日期格式

8. 选择“故障现象说明”输入框，在右侧属性页签中，单击，为输入框绑定“basicInfo”下的“***HW***__faultPhenomenon__CST”字段。
9. 参考上一步，为其他组件绑定数据字段，组件名及对应字段，如表2-21所示。

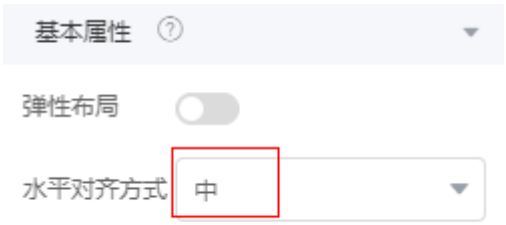
表 2-21 组件名及对应字段

标签名	绑定的字段名	组件类型
客户要求工程师到达现场时间	***HW***__arriveTime__CST	日期选择框
客户要求故障恢复时间	***HW***__recoveryTime__CST	日期选择框
客户接口人	***HW***__customer__CST	输入框

步骤14 组装“按钮”区域。

1. 选择页面最下方“容器”，设置“水平对齐方式”为“中”，并拖入2个“按钮”组件。

图 2-178 设置容器对齐方式



2. 修改第1个“按钮”的“显示名称”为“提交”。
3. 修改第2个“按钮”的“显示名称”为“取消”，“类型”为“默认按钮”。

步骤15 单击页面上方的，保存页面。

----结束

2.7.3.2.3 添加页面事件

通过在“***HW***__createWorkOrder”页面上，定义页面事件，实现将工单信息存入到工单对象中。

实现“根据设备编码自动加载工单其他信息”

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 单击打开“***HW***__createWorkOrder”页面。

步骤3 添加页面事件代码。

1. 在“设计视图”中，选中最外层的“页面”，也可以直接单击组件导航，快速定位。
2. 在右侧“事件”页签中，单击“加载”后的“+”，为页面添加事件代码。
3. 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

```
$model.ref('equipmentOptions').setData({inputParam:{}});
$model.ref('equipmentOptions').run().then(function(response){
if(response.equipList){
        $model.ref("equipmentList").setData(response.equipList);
}
});
```

4. 单击“创建”，退出事件编排窗口。

----结束

实现“显示设备详细信息”

通过定义“基本信息”区域“设备名称”下拉框的“数据改变”事件，可以实现根据所选设备，在“设备详情”区域显示设备信息。

步骤1 在“***HW***__createWorkOrder”页面中，选中“设备名称”下拉框。

图 2-179 添加“数据改变”事件代码

步骤2 在右侧“事件”页签中，单击“数据改变”后的“+”。

步骤3 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

须知

脚本中加粗斜体内容请替换为实际的命名空间前缀。

```
let equipment = $model.ref("basicInfo").getData();
let equipmentId = equipment.HW__equipmentId__CST;
let queryEquipParam = {"id" : equipmentId};
$model.ref('equipmentOptions').setData({inputParam: queryEquipParam});
$model.ref('equipmentOptions').run().then(function (response) {
    if (response.equipment) {
        $model.ref("equipmentInfo").setData(response.equipment);
    }
});
```

步骤4 单击“创建”，退出事件编排窗口。

----结束

实现“生成工单”

通过定义“提交”按钮的“点击”事件，实现生成工单的能力。定义“取消”按钮的“点击”事件，返回工单列表页面（客服人员）。

步骤1 定义“提交”按钮“点击”事件。

1. 在“*HW*__createWorkOrder”页面的“设计视图”下，选中“提交”按钮。
2. 在右侧“事件”页签中，单击“点击”后的“+”。
3. 在“添加动作”弹窗的“自定义动作”中，输入如下事件代码。

须知

脚本中加粗斜体内容请替换为实际的命名空间前缀。

```
// 配置页面的bpm参数bp.name，通过submitTask方法启动BPM并提交工单数据workOrderData到BPM
context.$page.params["bp.name"] = "HW__WorkOrderBpm";
let basicInfo = $model.get('basicInfo').getData();
basicInfo.HW__workOrderId__CST = basicInfo.HW__type__CST + "_" + new Date().getTime();
let workOrderData = {
    "workOrderData": basicInfo
};
context.$bp.submitTask(workOrderData).then(function (resp) {
```

```
    context.$page.loadStdPage('HW__workOrderList');
});
```

4. 单击“创建”，退出事件编排窗口。

步骤2 定义“取消”按钮的“点击”事件。

1. 在“设计视图”中，选中“取消”按钮。
2. 参考“提交”按钮，给“取消”按钮定义以下代码事件

```
//返回工单列表页面
context.$page.loadStdPage('HW__workOrderList');
```

3. 单击“创建”，退出事件编排窗口。

----结束

验证

页面组装及事件添加完成后，需要在预览页面，初步检查页面预览效果，验证页面的数据绑定及事件代码正确性。当前工单流转相关的BPM及工单列表尚未创建，因此暂不验证“保存”、“取消”按钮。

步骤1 在“***HW***__createWorkOrder”页面中，单击界面上方的  ，进入预览页面。

系统会弹出“***HW***__createWorkOrder”预览页面。

步骤2 在“基本信息”区域的“设备名称”下拉框中，选中一个设备名称，检查“设备详情”区域是否显示了对应的设备详情。显示正确，则说明组件与模型的绑定，以及下拉框事件执行正确。如不显示，则需要检查相关事件代码。

步骤3 检查“工单类型”、“优先级”下拉框中，是否有值，显示正确则说明下拉框属性设置正确。

----结束

2.7.3.3 开发“客服人员查询工单”功能

2.7.3.3.1 创建“查询工单”脚本及公共接口

本节中的后台逻辑主要是通过脚本的形式，实现对工单记录的查询，然后将查询脚本封装成一个公共接口，供页面调用。

工单列表（客服人员）页面中，为了实现查询并显示已有工单信息功能，需要开发“查询工单”脚本和对应公共接口。

创建“查询工单”脚本

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“queryWorkOrder”，单击“添加”。

系统实际创建的脚本名称为“***HW***__queryWorkOrder”，“***HW***__”前缀由租户命名空间namespace决定。新建创建的脚本，默认是当前用户锁定状态，可以进行编辑保存等操作。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
import * as context from 'context';
import * as db from 'db';
import * as decimal from 'decimal';//导入decimal数据类型相关的标准库

@useObject(['HW__WorkOrder__CST'])

@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String' })
    title: string;
    @action.param({ type: 'String' })
    status: string;
    @action.param({ type: 'String' })
    createdBy: string;
    @action.param({ type: 'Boolean' })
    isFME: boolean;
    @action.param({ type: 'Number', min: 0 })
    start: decimal.Decimal;//分页信息，表示从第几条数据开始查询
    @action.param({ type: 'Number', min: 0 })
    limit: decimal.Decimal;//分页信息，表示一次查询几条数据
}

@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any', label: 'object', isCollection: true })
    workOrderList: object[];
    @action.param({ type: 'String' })
    total: string;//总共查到几条数据
}

@action.object({ type: "method" })
export class QueryWorkOrder {
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public queryWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let s = db.object('HW__WorkOrder__CST');    //获取HW__WorkOrder__CST这个Object的操作实例

            //condition是db标准库queryByCondition方法的入参（查询条件）
            let condition = {
                "conjunction": "AND",
                "conditions": []
            };
            //基本查询条件
            condition.conditions.push({
                "field": "id",
                "operator": "isnotnull",
            });
            condition.conditions.push({
                "field": "id",
                "operator": "ne",
                "value": ""
            });
            //按title查询
            if (input.title && input.title != "") {
                condition.conditions.push({
                    "field": "HW__title__CST",//与对象中的字段名保持一致
```

```
                "operator": "contains",
                "value": input.title
            });
        }
        //按status查询
        if (input.status && input.status != "") {
            condition.conditions.push({
                "field": "HW__status__CST",//与对象中的字段名保持一致
                "operator": "eq",
                "value": input.status
            });
        }
        //按title查询
        if (input.createdBy && input.createdBy != "") {
            condition.conditions.push({
                "field": "createdBy",//与对象中的字段名保持一致
                "operator": "eq",
                "value": input.createdBy
            });
        }
        if (input.isFME) {
            condition.conditions.push({
                "field": "HW__assignedFme__CST",
                "operator": "eq",
                "value": context.getUserName()
            });
        }
        let option = {
            "options": {
                "orderby": [{
                    "field": "createdDate",
                    "order": "desc"
                }]
            }
        };
        //如果有分页
        if (input.start && input.limit) {
            let start = decimal.toNumber(input.start);//将decimal类型转换为接口需要的number类型
            let limit = decimal.toNumber(input.limit);
            option.options['limit'] = limit;
            option.options['skip'] = start;
        }
        let workOrderList = s.queryByCondition(condition, option);
        for (let i in workOrderList) {
            if (workOrderList[i].HW__status__CST == "关闭") {
                workOrderList[i].isDeal = true;
            }
        }
        out.workOrderList = workOrderList;
        //调用查询符合condition条件的数据总数的接口
        out.total = s.count(condition) + "";
    } catch (error) {
        console.error(error.name, error.message);
        context.setError(error.name, error.message);
    }
    return out;
  }
}
```

步骤5 单击编辑器上方的，保存图标。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的，执行脚本。
2. 在界面底部，查询脚本可以不提供输入参数，直接单击测试窗口右上角执行图标。
   执行成功，会在“输出”页签返回查询结果。

图 2-180 输出结果示例

步骤7 测试成功，单击编辑器上方的，启用发布脚本。

----结束

创建“查询工单”公共接口

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

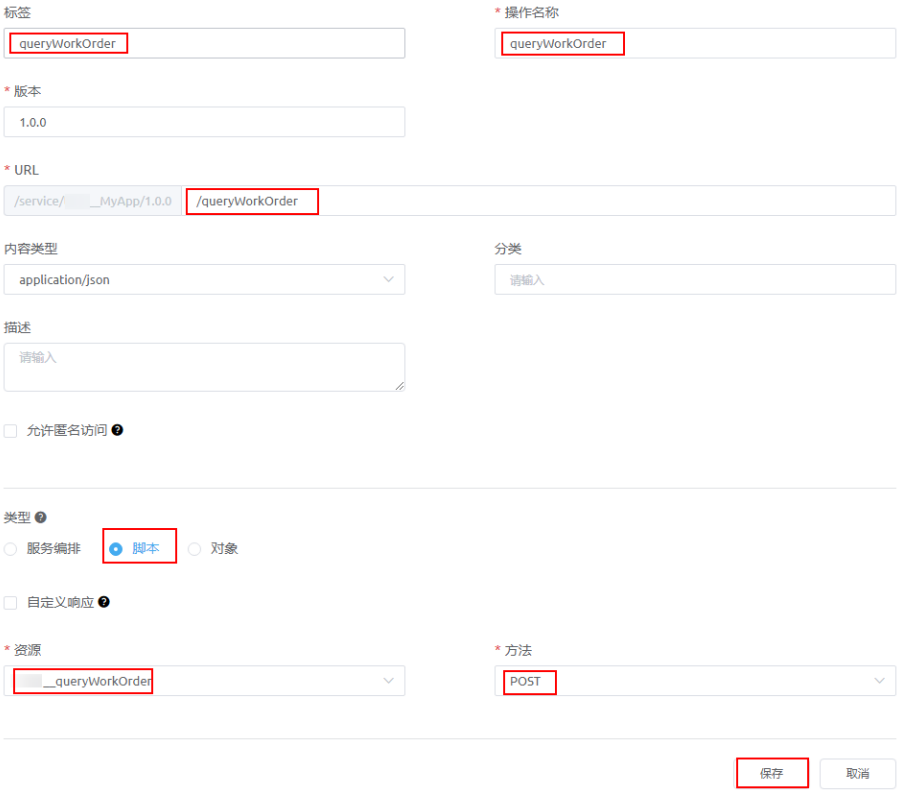
步骤2 单击页面下方的“服务”，进入公共接口创建页面。

步骤3 单击“新建”，创建“查询工单”脚本“***HW***__queryWorkOrder”的公共接口。

设置接口参数信息：操作名称为“queryWorkOrder”，版本为“1.0.0”，URL为“/queryWorkOrder”，“类型”选择“脚本”，“资源”为“***HW***__queryWorkOrder”，方法为“POST”，然后单击“保存”。

其中，“***HW***__”前缀由租户命名空间namespace决定，请根据实际情况进行选择。

图 2-181 新建 queryWorkOrder 接口



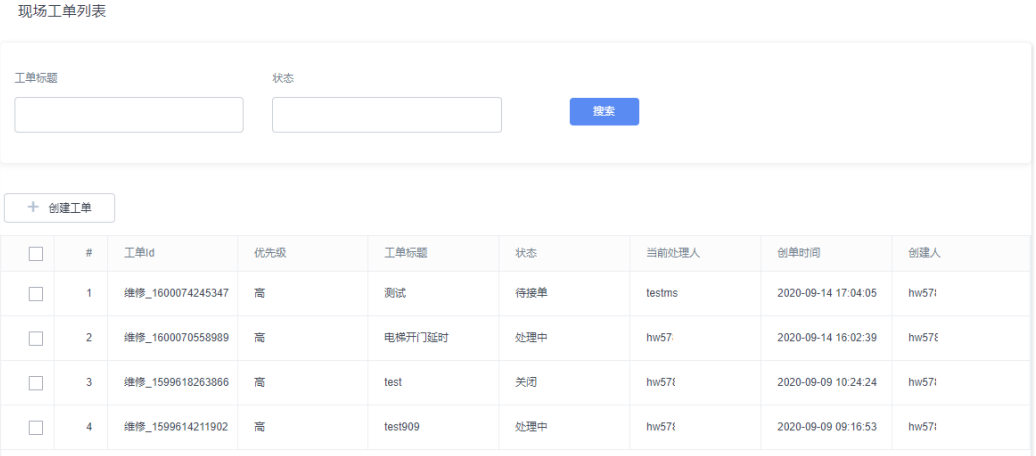
----结束

2.7.3.3.2 组装“工单列表（客服人员）”页面

“工单列表（客服人员）”页面与查询设备列表页面类似，包含工单查询条件、工单列表，以及创建工单的入口，用标准页面功能实现。

如图2-182所示，工单列表（客服人员）页面主要是一个表格、“创建工单”按钮及表格查询区域构成。

图 2-182 工单列表（客服人员）



操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”中，在鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“workOrderList”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 定义模型“queryWorkOrder”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加服务模型，模型名称“queryWorkOrder”，单击“下一步”。
3. “选择服务类型”为“公共接口”，“项目”为“设备维修管理系统”，“选择服务”为“queryWorkOrder”，单击“下一步”，再单击“确定”。
4. 单击页面上方的  ，保存模型。

步骤5 定义模型“workOrderInstance”。

1. 在“模型视图”中，单击“新增模型”。
2. 来源选择“自定义”，模型名称“workOrderInstance”，单击“下一步”。
3. 单击模型名后的“新增节点”，新增表2-22所示的节点，然后单击“下一步”，再单击“确定”。

   以下字段要与“工单对象”中的字段一致，其中加粗斜体内容部分要替换为当前账号对应的命名空间前缀。

   表 2-22 新增节点

字段名称	字段类型	字段描述
id	Text	记录ID
HW__workOrderId__CST	Text	工单Id
HW__priority__CST	Text	优先级
HW__title__CST	Text	工单标题
HW__status__CST	Text	状态
HW__assignedFme__CST	Text	当前处理人
createdDate	DataTime	创单时间
HW__createdBy__CST	Text	创建人

4. 单击页面上方的  ，保存模型。

步骤6 单击“设计视图”，从“模型视图”切换到“设计视图”。

步骤7 拖拽页面标题。

1. 从基本组件列表区拖拽一个“标题”组件到“页面内容”中。

2. 在右侧“属性”页签中，将“标题内容”修改为“现场工单列表”。
3. 单击页面上方的，保存页面标题。

步骤8 从组件列表区拖拽一个“表格”组件到“标题”的下方。

步骤9 绑定表格数据模型，并设置查询结果区域。

1. 在“设计视图”中，选中“表格”，单击右侧“属性”页签，如图2-183所示。

图 2-183 为表格绑定数据模型

2. 在“选择模型”对话框中，选中“workOrderInstance”模型，单击“确定”。绑定模型后，系统自动将模型的所有字段添加为查询结果列。
3. 单击上图中每个字段后的，修改列标题等字段属性。

   以“工单ID”为例，其属性配置如图2-184所示，各个字段及修改后的列名如表2-23所示。

图 2-184 设置“工单 ID”列的属性

表 2-23 修改各字段的显示属性

字段名	列标题	备注
id	记录ID	启用“隐藏”属性，即设置为true
HW__workOrderId__CST	工单ID	-

字段名	列标题	备注
HW__priority__CST	优先级	-
HW__title__CST	工单标题	-
HW__status__CST	状态	-
HW__assignedFme__CST	当前处理人	-
createdDate	创单时间	-
HW__createdBy__CST	创建人	-

4. 单击页面上方的，保存设置。

步骤10 添加“创建工单”按钮。

1. 选中“表格”，单击右侧“属性”页签中“表格区块”中“工具栏”后的“添加”按钮。

图 2-185 增加查询条件区域

2. 删除工具栏中多余的按钮，只保留“新增行”。
3. 修改“新增行”的“显示名称”为“创建工单”。
4. 单击页面上方的，保存设置。

步骤11 创建查询条件区域。

1. 单击“模型视图”，切换到模型视图，在表格模型“table_***0***_condition”后，单击编辑图标。

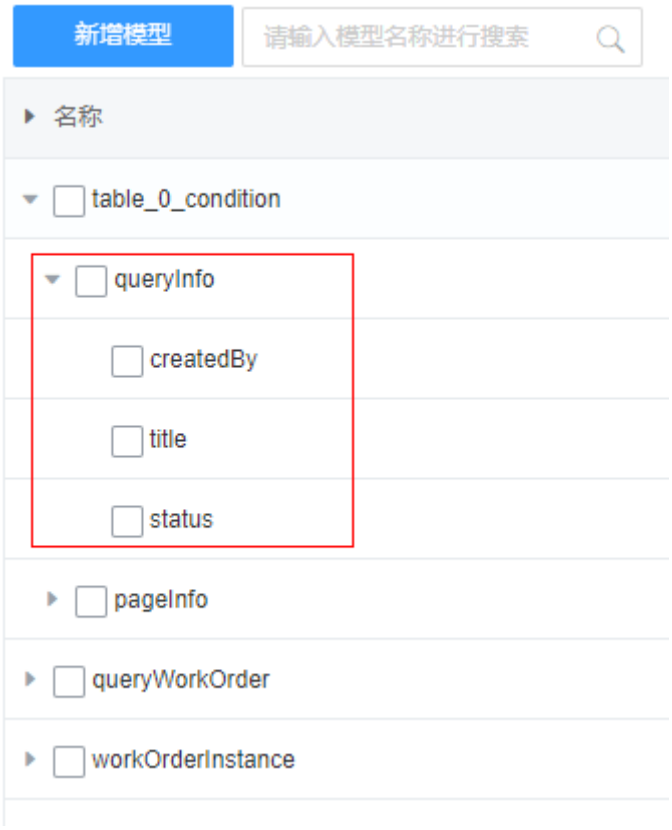
说明

表格组件拖入后，页面将自动生成一个表格模型。“table_***0***_condition”即是在前面拖入的表格对应的表格模型，模型名可能跟拖入顺序及个数有关，一般情况下模型名称为“table_***0***_condition”，如果名称不一致，请根据实际情况修改。

图 2-186 编辑表格模型

2. 在表格模型“table_*0*_condition”中，单击“新增节点”，添加一个“queryInfo”节点，“数据类型”为“Any”，然后在“queryInfo”下，再单击“新增节点”添加3个子节点“createdBy”、“ title”、“status”。

图 2-187 创建完成后的模型节点



3. 设置完成后，单击“下一步”，再单击“确定”，保存设置。
4. 单击“设计视图”，切换到设计视图，从左侧组件列表中，拖一个“表单”组件到“表格容器”的最上部（表格容器内部最上部），在“元数据表单配置向导”弹窗中，单击“取消”。

图 2-188 向表格容器拖入表单

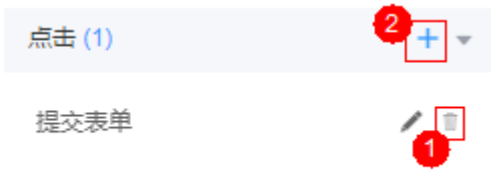
5. 选中表单，在属性中“数据绑定”下，单击，为表单绑定“table_*0*_condition”中的“queryInfo”节点，并在提示弹窗中，单击“绑定并生成表单”，单击“确定”。

6. 在表单中，分别选中并删除“createdBy”、“重置”按钮。
7. 调整“title”、“status”、“保存”为1行，并删除多余分栏及容器，调整后，如图2-189所示。

图 2-189 调整后表单

8. 修改表单中输入框属性。
   - 修改“title”的“标签”为“工单标题”。
   - 修改“status”的“标签”修改为“状态”。
9. 选中“保存”按钮，修改“显示名称”为“搜索”，设置单击“事件”页签，再单击，删除“提交表单”事件，然后单击“点击”后的“+”，进入事件编排页面，在“自定义动作”中，输入以下事件代码，单击“创建”。

图 2-190 修改按钮事件代码



须知

脚本中**红色内容**请替换为实际的对象名、字段名、页面名、组件ID。其中“table_0_condition”为当前表格组件的ID号，中间数字默认为“0”，如果有多次修改或创建该ID号会变化，选中表格后，在组件树上可以查看实际组件ID。

```
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
let queryInfo = $model.ref('table_0_condition').getData().queryInfo;
if (!queryInfo) {
    queryInfo = {};
}
let queryData = {
    "title": queryInfo.title,
    "status": queryInfo.status,
    "createdBy": queryInfo.createdBy,
    "start": 0,
    "limit": pageInfo.pageSize
};
$model.ref('queryWorkOrder').setData({ inputParam: queryData });
$model.ref('queryWorkOrder').run().then(function (data) {
    $model.ref('workOrderInstance').setData(data.workOrderList);
    pageInfo.total = parseInt(data.total);
    $model.ref('table_0_condition').setData({ "pageInfo": pageInfo });
}).catch(function (error) {
    console.log('error is', error);
});
```

10. 选中“搜索”按钮所在的栏，在右侧“样式”下的“布局”中，设置上方外边距为“30”，使“搜索”按钮上下居中显示。

图 2-191 设置搜索按钮布局样式（位置）

查询条件区域创建后，如图2-192所示。

图 2-192 查询区域

11. 单击页面上方的，保存设置。

步骤12 添加页面事件代码。

1. 在“设计视图”中选中最外层的“页面”，在右侧“事件”页签中，单击“加载”后的“+”，进入编辑动作页面。
2. 在“自定义动作”中，输入如下脚本代码。

```
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
let queryData = {
    "start": (pageInfo.curPage - 1) * pageInfo.pageSize,
    "limit": pageInfo.pageSize
};
$model.ref('queryWorkOrder').setData({ inputParam: queryData });
$model.ref('queryWorkOrder').run().then(function (data) {
    $model.ref('workOrderInstance').setData(data.workOrderList);
    pageInfo.total = parseInt(data.total);
    $model.ref('table_0_condition').setData({ "pageInfo": pageInfo });
}).catch(function (error) {
    console.log('error is', error);
});
```

3. 单击“创建”，退出事件编排窗口。

步骤13 添加“创建工单”按钮跳转事件，跳转事件功能是将页面跳转到“生成工单”页面。

1. 选中“创建工单”按钮，在右侧“事件”页签中，单击“新增行”后的，进入事件编排页面。
2. 在“自定义动作”中，输入以下事件代码。

```
//打开创建工单页面，需要根据实际页面名称修改
context.$page.load('/besBaas/page#/HW__createWorkOrder')
```

3. 单击“创建”，退出事件编排窗口。

步骤14 单击页面上的，保存页面。

----结束

2.7.3.4 验证

“工单列表（客服人员）”和“生成工单”页面开发完成后，需要验证两个页面间的关联跳转事件，以及相关页面布局样式。

操作步骤

步骤1 在“***HW***__workOrderList”页面中，单击界面上方预览图标。

系统会弹出预览页面。

步骤2 查看页面中页面布局、样式是否符合预期。

步骤3 查看当前表格中的工单记录。当前表格中仅有一条测试数据，此数据是在**生成工单**脚本中输入的一条测试数据。

图 2-193 查看工单记录

步骤4 验证搜索功能。

1. 在“工单标题”中，输入“电梯维修”，进行搜索，因为当前没有该标题的工单，因此搜索结果为空，然后再输入“电梯无法关门”，如果显示该工单记录，则说明搜索功能正确。
2. 在“状态”中输入“待处理”，然后单击“搜索”，当前暂时没有“待处理”工单，因此搜索结果为空，再输入“待派单”后，单击“搜索”，则显示已存在工单，则说明搜索功能正常。

步骤5 验证页面跳转事件。

单击“创建工单”，验证是否跳转到“生成工单”页面。如果页面跳转到工单列表页面，则说明验证成功。

因当前生成工单页面相关的BPM尚未创建，这里还无法验证创建工单功能。

----结束

2.7.4 开发“派单员派发工单”功能

“派单员派发工单”功能包含工单列表（派单员）页面、派单对话框两个页面。

学习地图

如**图2-194**所示，通过本章的学习和实践，您将进一步了解“标准页面”的能力，包括：

- 弹出对话框
- 标准页面模板
- 脚本

图 2-194 学习地图

2.7.4.1 创建“查询维修人员”脚本

当派单员选择派单的下一步接单人时，需要先查询出，维修人员的列表，然后才能将流程走到维修人员的名下，因此需要创建一个“查询维修人员”的脚本逻辑。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“queryWorker”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

```
import * as context from 'context';
import * as db from 'db';

//使用数据对象PortalUser(业务用户)
@useObject(['PortalUser'])

@action.object({ type: "param" })
export class ActionInput {
}

@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any', label: 'object', isCollection: true })
    userList: object[];
}

@action.object({ type: "method" })
export class QueryWorker {
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public queryWorker(input: ActionInput): ActionOutput {
        let out = new ActionOutput();
        let portalUserObject = db.object('PortalUser');

        //查询用户类型为ms（维修人员）的用户。（注册用户的脚本中设置了用户类型）
        let portalUsers = portalUserObject.queryByCondition({
            "conjunction": "AND",
            "conditions": [{
                "field": "userType",
```

```
            "operator": "eq",
            "value": "ms"
        }]
    });

    //将查询结果转换为选项列表的形式（id和name）。
    let selectValue = portalUsers.map(function (v, i, a) {
        return {
            'value': {
                "id": v['id'],
                "name": v['usrName']
            },
            'display': v['usrName']
        }
    });

    //当前登录的用户为平台用户（非业务用户），则加入到维修人员列表中，用于测试。
    if (context.getUserType() == context.UserType.User) {
        selectValue.push({
            'value': {
                "id": context.getUserId(),
                "name": context.getUserName()
            },
            'display': context.getUserName()
        });
    }

    out.userList = selectValue;

    return out;
  }
}
```

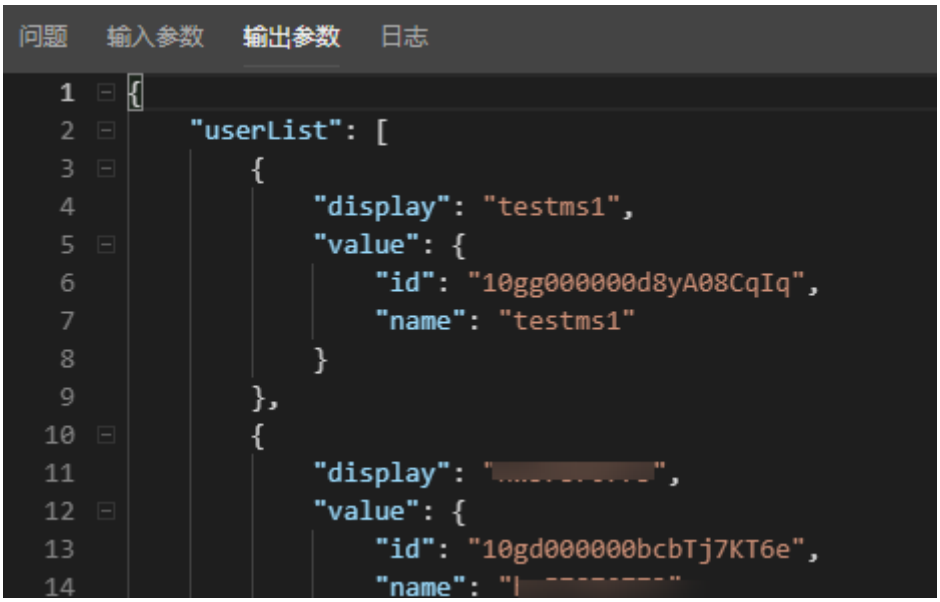
步骤5 单击编辑器上方的，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的，执行脚本。
2. 在界面底部，单击测试窗口右上角执行图标。

   执行成功，会在“输出”页签返回查询结果，请记录“name”值，此参数在**创建“派单功能”脚本**会作为输入参数使用。

图 2-195 输出查询出的维修人员信息



步骤7 测试成功，单击编辑器上方的，启用发布脚本。

----结束

2.7.4.2 创建“派单功能”脚本

当派单员选择派单的下一步接单人时，需要修改工单状态及处理人，因此需要创建一个实现“派单功能”的脚本逻辑，该脚本逻辑不是页面直接调用，而是通过后端BPM调用运行的。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“dispatchWorkOrder”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 在代码编辑器中，插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于派发工单
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import * as date from 'date';
//定义入参结构，入参包含1个参数：工单状态的修改信息，为必填字段
@action.object({ type: "param" })
```

```
export class ActionInput {
    @action.param({ type: 'Any', required: true, label: 'object' })
    transInfo: any;
}
//定义出参结构，出参包含2个参数，workOrder的记录id和派发后的责任人
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
    @action.param({ type: 'String' })
    assignedFme: string;
}
//使用数据对象HW__WorkOrder__CST
@useObject(['HW__WorkOrder__CST'])
@action.object({ type: "method" })
export class DispatchWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public dispatchWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let transInfo = input.transInfo;    //将入参赋值给transInfo变量，方便后面使用
            let s = db.object('HW__WorkOrder__CST');    //获取HW__WorkOrder__CST这个Object的操作实例

            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "HW__workOrderId__CST",
                    "operator": "eq",
                    "value": transInfo['HW__workOrderId__CST']
                }]
            };
            //查找workOrderId所代表的工单信息
            let workOrder = s.queryByCondition(condition);
            workOrder[0]['HW__status__CST'] = '待接单';
            workOrder[0]['HW__assignedFme__CST'] = transInfo["HW__assignedFme__CST"].name;
            let isUpdated = s.updateByCondition(condition, workOrder[0]);
            if (isUpdated) {
                out.id = workOrder[0]['id'];
                out.assignedFme = transInfo["HW__assignedFme__CST"].id;
            } else {
                error.name = "WOERROR";
                error.message = "派发工单失败！";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方的  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的  ，执行脚本。
2. 在界面底部输入测试数据，单击测试窗口右上角  执行图标。

   脚本中加粗斜体内容请替换为实际的对象名、字段名。 "name":"***test***"为当前租户名，可以在上一节的**输出参数**中查询。

   ```
   {
     "transInfo": {
       "***HW***__workOrderId__CST": "WD0000123456",
       "***HW***__assignedFme__CST":{
   ```



```
        "id":"",
        "name":"test"
      }
    }
}
```

执行成功，会在“输出”页签返回查询结果。

图 2-196 返回结果

步骤7 测试成功，单击编辑器上方的，启用发布脚本。

----结束

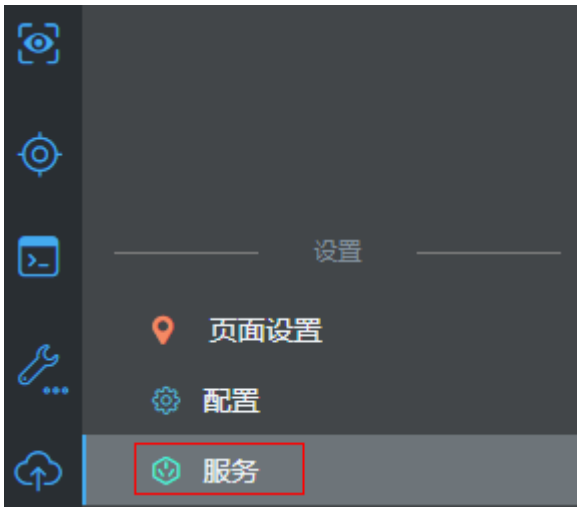
2.7.4.3 创建公共接口

参考之前创建公共接口的步骤，创建“查询维修人员”脚本对应的公共接口。

操作步骤

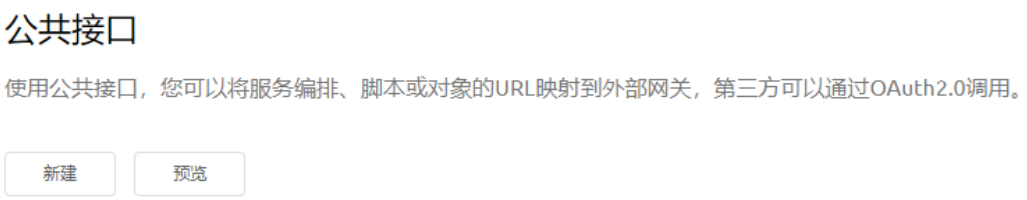
步骤1 在设计视图下，单击下方的“服务”，进入公共接口创建页面。

图 2-197 创建公共接口入口



步骤2 在公共接口页面，单击“新建”。

图 2-198 公共接口创建



步骤3 创建“查询维修人员”、“派单功能”脚本对应的公共接口，详细接口信息如表2-24所示。

说明

加粗斜体内容请替换为实际命名空间前缀。

表 2-24 公共接口

设置操作	版本	URL	方法	类型	资源
queryWorker	1.0.0	/queryWorker	GET	脚本	***HW***__queryWorker
dispatchWorkOrder	1.0.0	/dispatchWorkOrder	POST	脚本	***HW***__dispatchWorkOrder

----结束

2.7.4.4 组装“派单”对话框

“派单”对话框负责指定下一环节的工单状态和处理责任人，用标准页面功能实现。

派单界面的大致构想如图2-199所示，其中“工单ID”是为了传递维修工单信息。

图 2-199 派单员派单对话框

页面模型负责与页面组件交互，传递派单需要的工单ID、工单状态、下一环节处理人等信息。结合页面需求，可知需要创建如下模型：

表 2-25 模型分析

模型名称	作用	详细定义
transInfo	保存派单的参数。	自定义模型，包含的计算节点如下，这些节点与派单接口的输入参数名称一一对应。 • ***HW***__workOrderId__CST：工单ID • ***HW***__assignedFme__CST：下一环节处理人 **说明** 加粗斜体内容请替换为实际命名空间前缀。
workerOptions	查询系统中的工程师信息，下一环节处理的可选项。	服务模型，与公共接口queryWorker关联。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”中，鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“workOrderDispatch”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 定义模型“transInfo”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加自定义模型，模型名称“transInfo”，单击“下一步”。
3. 为模型添加节点“***HW***__workOrderId__CST”（字段类型Text）、“***HW***__assignedFme__CST”（字段类型Any），单击“下一步”，再单击“确定”，加粗斜体内容请替换为实际命名空间前缀。
4. 单击页面上方的，保存模型。

步骤5 定义模型“workerOptions”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加服务模型，模型名称“workerOptions”，单击“下一步”。
3. “项目”选择“设备维修管理系统”，并为模型关联API“queryWorker”，单击“下一步”，再单击“确定”。
4. 单击页面上方的，保存模型。

步骤6 拖拽页面组件。

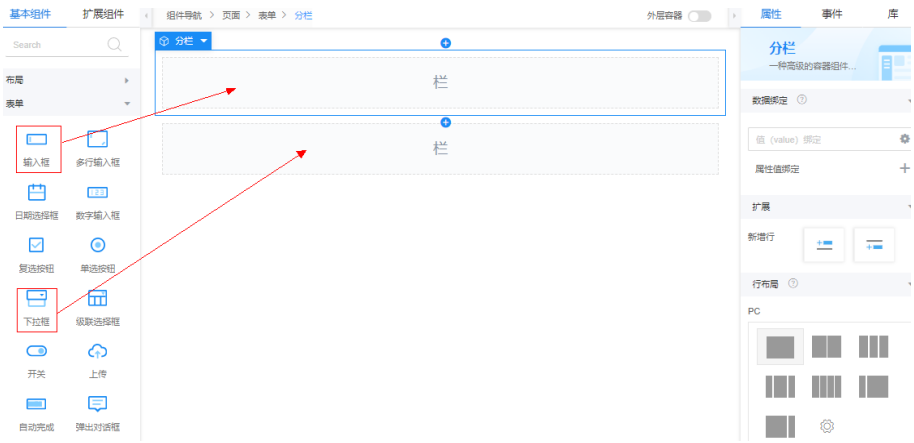
1. 单击“设计视图”，返回页面设计。
2. 将左侧基本组件区的“表单”拖拽到右侧页面中。
   因为当前还没有定义数据源，单击“取消”，创建一个空的表单控件。

3. 拖拽一个“分栏”到表单中。
4. 选中“分栏”，在右侧单击PC下的单行图标，修改分栏为1栏，再单击“新增行”的，修改为2个分栏，每个分栏中有1栏。

图 2-200 设置分栏为 1 栏

5. 在第1个分栏中拖入一个“输入框”，在第2个分栏中拖入一个“下拉框”。

图 2-201 拖入输入框、下拉框到分栏

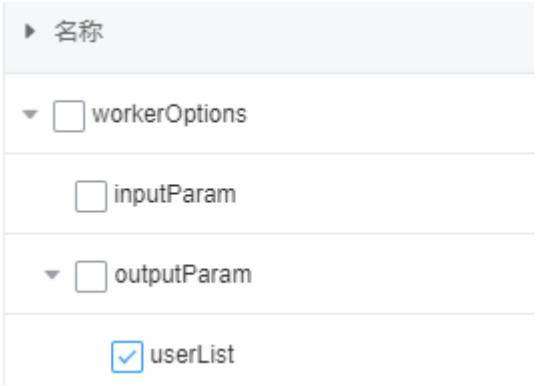


6. 选中“输入框”，在右侧属性面板中，单击，在选择模型弹窗中，选择“transInfo”下的“***HW***__workOrderId__CST”字段，并修改其“标签”为“工单ID”。
7. 选择下拉框进行以下设置。

a. 在右侧属性面板中，单击，在选择模型弹窗中，选择“transInfo”下的“***HW***__assignedFme__CST”字段，修改“标题”为“选择工程师”。

b. 单击“属性值绑定”后的“+”，设置“属性”为“选项”，“模型字段”为“workerOptions.outputParam.userList”，即绑定属性值为服务对象的返回值。

图 2-202 绑定服务对象的返回值



c. 开启“弹层独立”。

图 2-203 启用弹层独立

8. 单击页面上方的，保存设置。

步骤7 定义页面事件代码。

1. 在“设计视图”中选中最外层的“页面”，在右侧“事件”页签中，单击“加载”后的“+”，进入编辑动作页面。
2. 在“自定义动作”中，输入如下脚本代码。

须知

红色内容请替换为实际命名空间前缀。

```
let workOrderId = Page.params.workOrderId;
$model.ref("transInfo").setData({ "HW__workOrderId__CST": workOrderId });
//查询维修人员列表，作为维修人员下拉框的可选值
$model.ref('workerOptions').setData({ inputParam: {} });
$model.ref('workerOptions').run();
```

3. 单击“创建”，关闭事件编排器，返回到页面。
4. 单击页面上方的，保存页面。

----结束

验证

单击界面上方的，进入预览页面，查看页面的展示效果。

因为这个页面需要从上一个页面获取工单信息，所以当前的预览效果只能看到空白“工单ID”以及“选择工程师”下拉框。

2.7.4.5 组装“工单列表（派单员）”页面

“工单列表（派单员）”页面与“工单列表（客服人员）”页面相比，大致功能一致，仅多了一个操作列，操作列中包含了派单员需要使用的派单图标。

因为两个页面很相似，所以为了快速创建，可以将“工单列表（客服人员）”页面保持为一个模板，再引用这个模板创建新的页面。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 使用“工单列表（客服人员）”页面创建新页面。

1. 打开“***HW***__workOrderList”，单击锁定页面。
2. 单击页面上方的，另存为页面。
3. 如图2-204所示，在弹出对话框中设置模板标签和名称为“dispatchWorkOrder”，单击“创建”。

   您可以看到新创建页面继承了原“工单列表（客服人员）”页面中的组件、模型和事件代码。

图 2-204 保存页面模板

步骤3 删除“创建工单”功能。

选中“创建工单”按钮所在的“数据表格-工具栏”，单击鼠标右键，单击“删除”，删除前请确认已选中“创建工单”按钮，以免误删其他组件。

步骤4 为查询结果增加操作列。

1. 在左侧“设计视图”中，选中“表格”，单击右侧“属性”页签“表格列”中的  ，添加操作列。

图 2-205 增加操作列

2. 单击上图中Operation1后的。
3. 在“属性配置”对话框中进行以下配置。
   a. 修改“列标题”为“操作”，单击“添加操作按钮”。
   b. 设置按钮“标签”为“派单”，“图标”选择“shuffle”，并在“禁用”中输入“$row.***HW***__status__CST!="待派单"”，即是当工单状态为“待派单”时，该按钮才可以使用。“***HW***__”请修改为实际命名空间前缀。

图 2-206 为操作列增加派单按钮

4. 单击图2-206中“派单”按钮的，再单击“添加动作”的“+”，进入事件编排中，在事件编排页面，添加如下事件代码，然后单击“创建”。

须知

脚本中加**红色内容**请替换为实际的页面名。

```
let _component = context.$component.current;
// 配置页面的bpm参数
context.$page.params["bp.name"] = "HW__WorkOrderBpm";
let currentRow = _component.$attrs.row;
let workOrderId = currentRow.HW__workOrderId__CST;
let instanceId = currentRow.HW__instanceId__CST;
let taskId = "";
let _model = $model.ref("workOrderInstance");
context.$dialog.popup(
  {
    title: '处理工单',    //弹出框标题
    page: 'HW__workOrderDispatch', // 弹出页面名称
    footerHide: false,
    width: 20, //宽，小于100为百分比，大于100为px
    height: 250, // 高px
    okText: '确定',
    cancelText: '取消',
```

```
    params: {
        "workOrderId": workOrderId
    }, //页面参数
    onCancel: () => {
        $model.ref("transInfo").setData({});
    },  // 取消时，回调
    onOk: () => {
        // 确定时，回调
        //debugger;
        let transInfo = { "transInfo": $model.ref("transInfo").getData() };
        let taskData = {
            "action": "complete",
            "variables": transInfo
        };

        // 获取csrf token
        context.$utils.getCSRFToken().then(function (token) {
            let url = '/u-route/baas/bp/v2.0/query/tasks?flag=activeTask&rootID=' + instanceId;
            fetch(url, {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'CSRF-Token': token
                }
            }).then(function (resp) {
                resp.json().then(function (data) {
                    taskId = data.result.Recs[0].id;
                    let url2 = '/u-route/baas/bp/v2.0/runtime/tasks/' + taskId;
                    fetch(url2, {
                        method: 'PUT',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                        body: JSON.stringify(taskData)
                    }).then(function (resp) {
                        context.$message.success('派发成功');
                        let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
                        let queryData = {
                            "start": (pageInfo.curPage - 1) * pageInfo.pageSize,
                            "limit": pageInfo.pageSize
                        };
                        $model.ref('queryWorkOrder').setData({ inputParam: queryData });
                        $model.ref('queryWorkOrder').run().then(function (data) {
                            $model.ref('workOrderInstance').setData(data.workOrderList);
                            pageInfo.total = parseInt(data.total);
                            $model.ref('table_0_condition').setData({ "pageInfo": pageInfo });
                        }).catch(function (error) {
                            console.log('error is', error);
                        });
                    }).catch(function (error) {
                        console.log('error is', error);
                    });
                });
            }).catch(function (error) {
                console.log('error is', error);
            });
        });
    }
  }
);
```

5. 单击“确定”，关闭“属性配置”对话框。
6. 单击页面上方的  ，保存设置。

步骤5 **检查页面**加载事件是否从模板复制过来。如果已经复制过来，则**忽略这步**。

1. 选中最外层的“页面”，在右侧“事件”页签中，单击“点击”后的“+”。
2. 在添加动作弹窗中，输入如下脚本代码。


```
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
let queryData ={
"start":(pageInfo.curPage -1)* pageInfo.pageSize,
"limit": pageInfo.pageSize
};
$model.ref('queryWorkOrder').setData({ inputParam: queryData });
$model.ref('queryWorkOrder').run().then(function(data){
    $model.ref('workOrderInstance').setData(data.workOrderList);
    pageInfo.total = parseInt(data.total);
    $model.ref('table_0_condition').setData({"pageInfo": pageInfo });
}).catch(function(error){
    console.log('error is', error);
});
```

3. 单击“创建”，退出事件编排窗口。

----结束

2.7.4.6 验证

因当前还未创建工单状态流转的BPM，因此派单功能还不能正常测试运行。本节只验证页面显示及跳转相关内容。

步骤1 在“***HW***__dispatchWorkOrder”页面中，单击界面上方的，进入页面预览，在页面预览中进行以下验证。

步骤2 查看页面显示：正常情况下，系统会显示客服人员创建的工单，且每条记录后都有派单图标，如果工单状态为“待派单”，则派单按钮为高亮可用状态，如果不是“待派单”状态，则按钮灰度不可用。

如果当前工单列表中没有“待派单”状态的工单，可以在**“生成工单”脚本**中输入一条测试数据，生成一条工单。

图 2-207 页面预览

步骤3 查看页面跳转。

选择一条“工单状态”为“待派单”的工单记录，单击“派单”按钮，查看是否弹出“处理工单对话框”，如果未跳转，请检查操作列“派单”按钮上的事件。

步骤4 在“处理工单对话框”中，查看是否显示“工单ID”，“选择工程师”下拉框是否显示正常，如果不正常，需要检查“派单”对话框相关事件代码及属性值绑定。

图 2-208 处理工单对话框

----结束

2.7.5 开发“维修工程师处理工单”功能

“维修工程师处理工单”功能包含“工程师查看待处理工单列表页面”和“处理工单对话框”两个页面。

2.7.5.1 创建“处理工单”脚本

“维修工程师处理工单”功能包含两个页面，一个是工程师查看待处理工单列表页面，一个是处理工单对话框页面。首先需要创建“处理工单”脚本。

操作步骤

步骤1 在经典版开发环境“首页 > 我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“modifyOrderStatus”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中加**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于派发工单
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import * as date from 'date';
//定义入参结构，入参包含1个参数：工单的状态信息，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'Object', required: true, label: 'object' })
    statusInfo: object;
}
//定义出参结构，出参包含2个参数，workOrder的记录id和当前状态
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
    @action.param({ type: 'String' })
    status: string;
}
//使用数据对象HW__WorkOrder__CST
@useObject(['HW__WorkOrder__CST'])
@action.object({ type: "method" })
export class ModifyOrderStatus {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public modifyOrderStatus(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let statusInfo = input.statusInfo;     //将入参赋值给statusInfo变量，方便后面使用
            let s = db.object('HW__WorkOrder__CST');    //获取HW__WorkOrder__CST这个Object的操作实例

            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "id",
                    "operator": "eq",
                    "value": statusInfo['id']
                }]
            };
            //查找workOrderId所代表的工单信息
            let workOrder = s.queryByCondition(condition);
            if (statusInfo['HW__status__CST'] == "接单") {
                workOrder[0]['HW__status__CST'] = "处理中";
            } else if (statusInfo['HW__status__CST'] == "拒单") {
                workOrder[0]['HW__status__CST'] = "待派单";
                workOrder[0]['HW__assignedFme__CST'] = "派单员";
            } else if (statusInfo['HW__status__CST'] == "关单") {
                workOrder[0]['HW__status__CST'] = "关闭";
            }
            let isUpdated = s.updateByCondition(condition, workOrder[0]);
            if (isUpdated) {
                out.id = workOrder[0]['id'];
                out.status = statusInfo['HW__status__CST'];
            } else {
                error.name = "WOERROR";
                error.message = "修改工单状态失败！";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方的  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的，执行脚本。
2. 在界面底部，输入如下示例，单击测试窗口右上角执行图标。
   “ceHg000000e0glLbDQ2K”则是派单员操作的当前工单记录的工单id，您可以在**派单功能**脚本的测试结果中获取一个id值。

```
{
   "statusInfo": {
      "id": "ceHg000000e0glLbDQ2K",
      "HW__status__CST": "接单"
   }
}
```

   执行成功，会在“输出”页签返回查询结果。

图 2-209 输出处理后工单信息

步骤7 测试成功，单击编辑器上方的，启用发布脚本。

----结束

2.7.5.2 创建“判断下一步状态”脚本

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“judgeNextStatus”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于判断下一步状态变化
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
//定义入参结构，入参包含1个参数：workOrder对象，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    id: string;
}
//定义出参结构，出参包含1个参数，workOrder的记录id
@action.object({ type: "param" })
export class ActionOutput {
```

```
    @action.param({ type: 'Any', label: 'Object', isCollection: true })
    statusList: object[];
}
//使用数据对象HW__WorkOrder__CST
@useObject(['HW__WorkOrder__CST'])
@action.object({ type: "method" })
export class JudgeNextStatus {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public judgeNextStatus(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let id = input.id;
            let s = db.object('HW__WorkOrder__CST');    //获取HW__WorkOrder__CST这个Object的操作实例
            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "id",
                    "operator": "eq",
                    "value": id
                }]
            };
            let workOrder = s.queryByCondition(condition);
            if (workOrder) {
                if (workOrder[0].HW__status__CST == "待接单") {
                    out.statusList = [
                        {
                            'value': "接单",
                            'display': "接单"
                        }, {
                            'value': "拒单",
                            'display': "拒单"
                        }];
                }
                if (workOrder[0].HW__status__CST == "处理中") {
                    out.statusList = [{
                        'value': "关单",
                        'display': "关单"
                    }];
                }
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方的  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的  ，执行脚本。
2. 在界面底部，输入如下输出参数报文，单击测试窗口右上角  执行图标，“ceHg000000e0glLbDQ2K”则是派单员操作的当前工单记录的工单id，您可以在**处理工单**脚本的测试结果中获取一个id值。

   ```
   {
       "id": "ceHg000000e0glLbDQ2K"
   }
   ```

   执行成功，会在“输出”页签返回查询结果。

图 2-210 返回下一步状态

步骤7 测试成功，单击编辑器上方的，启用发布脚本。

----结束

2.7.5.3 创建公共接口

参考之前创建公共接口的步骤，创建“查询维修人员”脚本对应的公共接口。

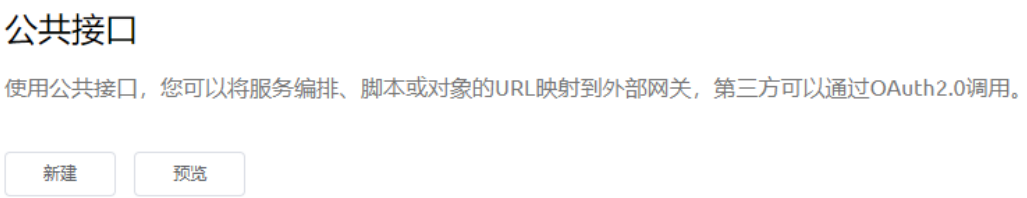
操作步骤

步骤1 在经典版应用开发页面的设计视图下，单击“服务”，进入公共接口创建页面。

图 2-211 服务入口

步骤2 单击“新建”，如下图所示。

图 2-212 公共接口创建



步骤3 创建“处理工单”、“判断下一步状态”脚本对应的公共接口，详细接口信息如表2-26所示。

表 2-26 公共接口

设置操作	版本	URL	方法	类型	资源
modifyOrderStatus	1.0.0	/modifyOrderStatus	POST	脚本	***HW***__modifyOrderStatus
judgeNextStatus	1.0.0	/judgeNextStatus	POST	脚本	***HW***__judgeNextStatus

----结束

2.7.5.4 组装“处理工单”对话框

“维修工程师处理工单”功能包含两个页面，一个是工程师查看待处理工单列表页面，一个是处理工单对话框。

“处理工单”对话框是一个手机端的Html5页面，负责指定下一环节的工单状态，是“待处理工单列表”页面中引用的对话框。

按照工程师在对话框中选定下一步流程状态，如图2-213所示。

图 2-213 现场工程师处理工单对话框

页面模型分析

页面模型负责与页面组件交互，传递处理工单需要的工单ID、工单状态等信息。结合页面需求，“处理工单”对话框页面，需要创建如下模型：

表 2-27 模型分析

模型名称	作用	详细定义
statusInfo	保存派单的参数。	自定义模型，包含的计算节点如下，这些节点与派单接口的输入参数名称一一对应。 ● id：工单ID ● ***HW***__status__CST：下一环节状态 **须知** **加粗斜体内容**请替换为实际命名空间前缀。
statusOptions	判断下一步状态。	服务模型，绑定公共接口“judgeNextStatus”，调用“judgeNextStatus”脚本，判断下一步状态。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“workOrderProcess”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击  进行锁定。

步骤4 单击页面右上部的  ，切换到手机端设计模式。

步骤5 定义模型“statusInfo”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加自定义模型，模型名称“statusInfo”，单击“下一步”。
3. 单击“新增节点”，为模型添加节点“id”和“***HW***__status__CST”字段，字段类型都采用默认的Text。单击“下一步”，再单击“确定”，加粗斜体内容请替换为实际命名空间前缀。
4. 单击页面上方的  ，保存模型。

步骤6 定义模型“statusOptions”。

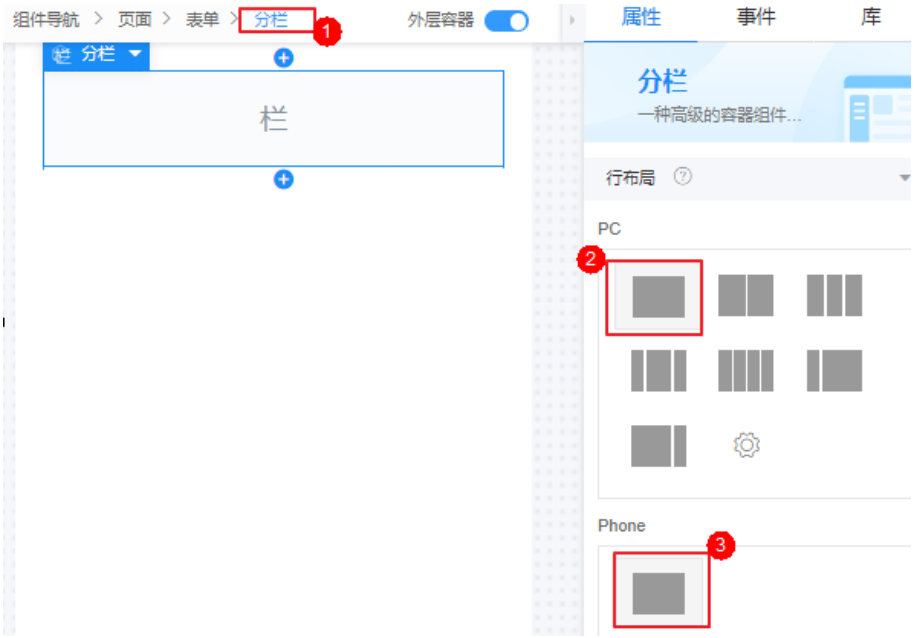
1. 在“模型视图”中，单击“新增模型”。
2. 添加服务模型，模型名称“statusOptions”，单击“下一步”，
3. “服务类型”选择“公共接口”，“选择项目”为“设备维修管理系统”，“搜索”中的接口选择“judgeNextStatus”，单击“下一步”，再单击“确定”。
4. 单击页面上方的  ，保存模型。

步骤7 拖拽页面组件。

1. 从“模型视图”切换到“设计视图”。
2. 从左侧基本组件列表区中，拖拽1个“表单”到“页面内容”中。

   因为当前还没有定义数据源，单击“取消”元数据表单配置向导弹窗，创建一个空的表单控件。
3. 拖拽1个“分栏”到“表单”。
4. 修改“分栏”为1栏（1行1列）。

图 2-214 设置分栏为 1 行 1 列



5. 拖拽1个“下拉框”到分栏的“栏”中，然后选中下拉框，在右侧属性面板中，对下拉框进行以下配置。

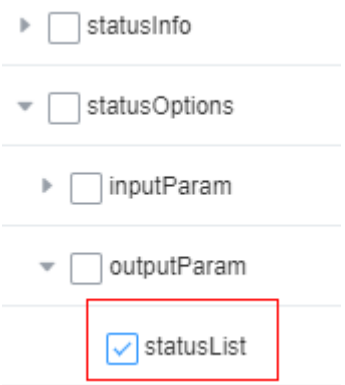
   a. 添加值绑定。

   单击“数据绑定”下“值绑定”的，在弹窗中选择“ statusInfo”下的“***HW***__status__CST”字段。

   b. 添加属性值绑定。

   单击“属性值绑定”的“+”，设置“属性”为“选项”，“模型字段”为“statusOptions”下“outputParam”节点的“statusList”。

   图 2-215 属性值绑定

   c. 修改“标签”为“选择下一步操作”，并设置“选项”为“user1，user1”、“user2，user2”。

图 2-216 设置选项值

d. 单击页面上方的，保存设置。

步骤8 定义页面事件代码。

1. 在“设计视图”中，选中最外层的“页面”。
2. 在右侧“事件”页签中，单击“加载”后的“+”。
3. 在添加动作弹窗中，输入如下事件代码。

```
let id = Page.params.id;
$model.ref("statusInfo").setData({ "id": id });
$model.ref('statusOptions').setData({ inputParam: { "id": id } });
$model.ref('statusOptions').run();
```

4. 单击“创建”，退出事件编排窗口。
5. 单击页面上方的，保存页面。

----结束

验证

单击界面上方的，进入预览页面，查看页面的展示效果。

因为这个页面需要从上一个页面获取工单信息，所以当前的预览效果只能看到“选择下一步”下拉框。

2.7.5.5 组装“待处理工单”页面

“待处理工单”页面是一个手机端的页面，用于显示现场工程师名下的待处理工单，将使用标准页面功能实现。

如图2-217所示工程师在手机端可以看到自己名下所有的待处理工单。系统默认优先显示待处理的工单。

图 2-217 “待处理工单”页面

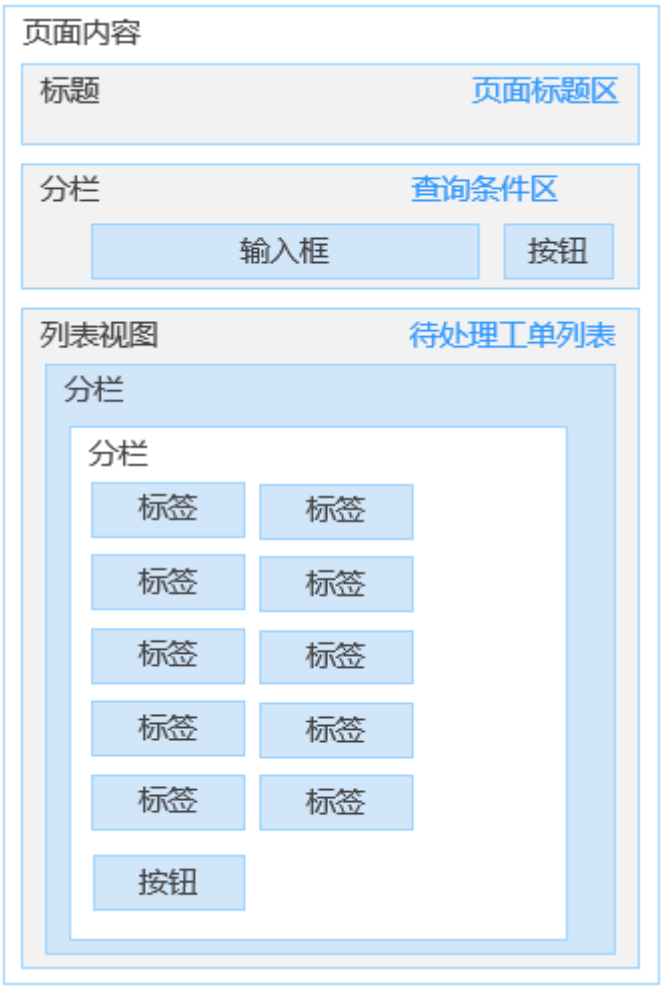
因当前页面组件较多，因此分3个大步骤开发页面，先拖拽页面的前端组件，再定义页面模型，进行模型与前端组件绑定，最后为页面添加相应的事件代码。

页面组件分析

分析图2-217的组成，可以将整个页面分割为3个区域：页面标题、查询条件和待处理工单列表区域，如图2-218所示。

待处理工单列表通过在列表视图中，使用2个“分栏”的嵌套，组装出待处理工单列表区。

图 2-218 页面组件分析



操作步骤

以下操作步骤只是给出的大致的拖拽方法，您可以根据需要，修改各组件的颜色、对齐方式、边距显示样式，使页面更美观。

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“workOrderListM”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 单击页面上方的Mobile图标，切换到手机端设计模式。

步骤5 组装“页面标题区”。

1. 从基本组件列表区，拖拽1个“标题”组件到“页面内容”。
2. 选中“标题”组件，在右侧“属性”页签，配置标题组件属性：
   a. 修改“标题内容”为“工单列表”。
   b. 在“样式”下的“高级设置”中，输入如下样式代码，即设置标题组件的背景色、水平居中对齐，文字颜色为白色。

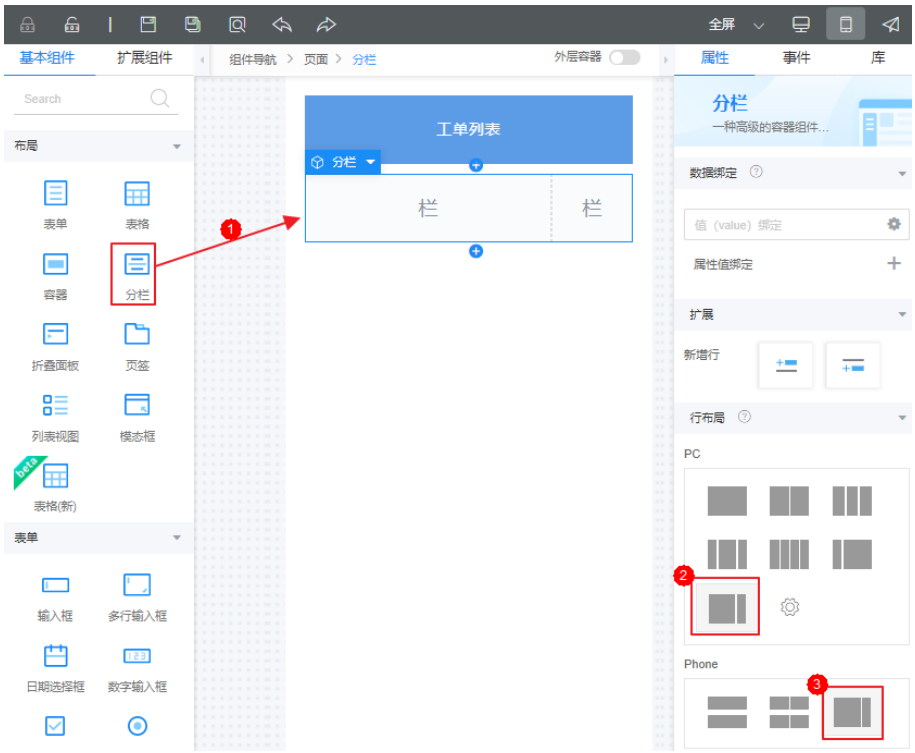
```
:root{
text-align:center;
background-color:#4a90e2;
color:#FFFFFF;
opacity:0.9;
}
```

图 2-219 高级设置

步骤6 组装“查询条件区”。

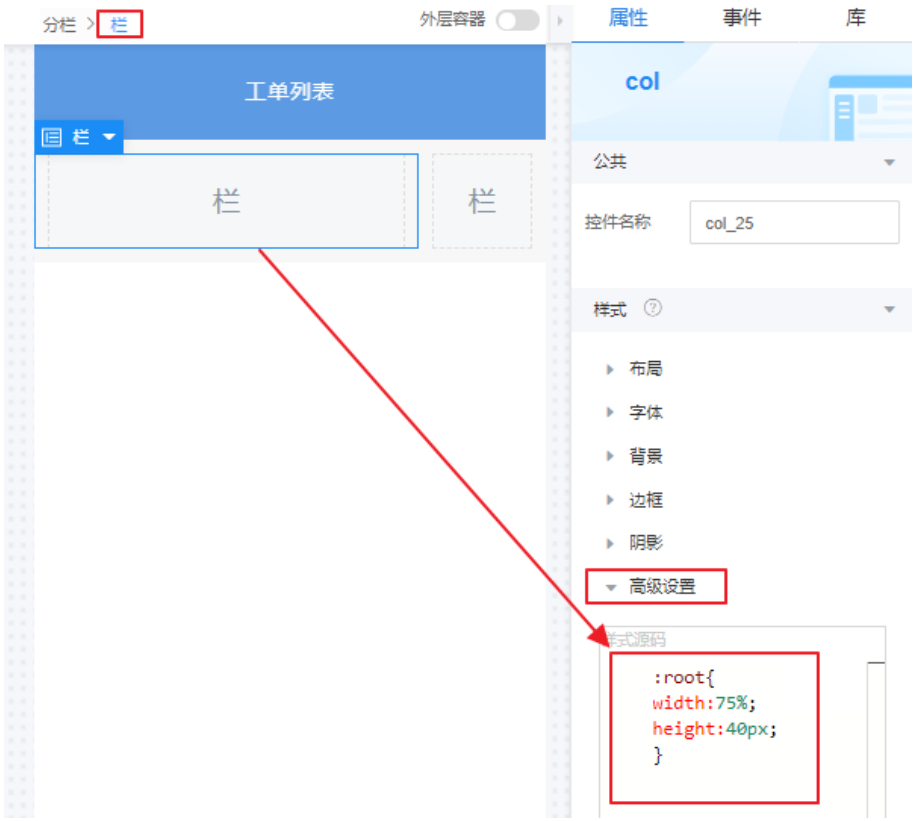
1. 拖拽1个“分栏”到“工单列表”之后，在右侧“行布局”下先单击PC中的2栏图标，再单击Phone下的2栏图标，将分栏设置为左右不均分的2栏，如下图所示。

图 2-220 拖入分栏组件并设置为 2 栏



2. 选中左侧“栏”，在“样式 > 高级设置”下输入样式代码，设置宽度为75%，高度为40px。

图 2-221 修改列宽度



```
:root{
width:75%;
height:40px;
}
```

3. 选中右侧“栏”，同样方式设置宽度为25%，高度为40px。

```
:root{
width:25%;
height:40px;
}
```

4. 拖拽1个“输入框”组件到“分栏”的左栏，修改其“占位符”为“请输入工单完整标题”。
5. 拖拽1个“按钮”组件到“分栏”的右栏，修改其“显示名称”为“查询”、“类型”为“主要按钮”。
6. 选择“请输入工单完整标题”所在的“栏”，设置“弹性布局”，并修改内边距为“10px”。

图 2-222 设置栏属性

步骤7 组装“待处理工单列表区”。

1. 在“分栏”之后，拖入1个“ 列表视图”组件。

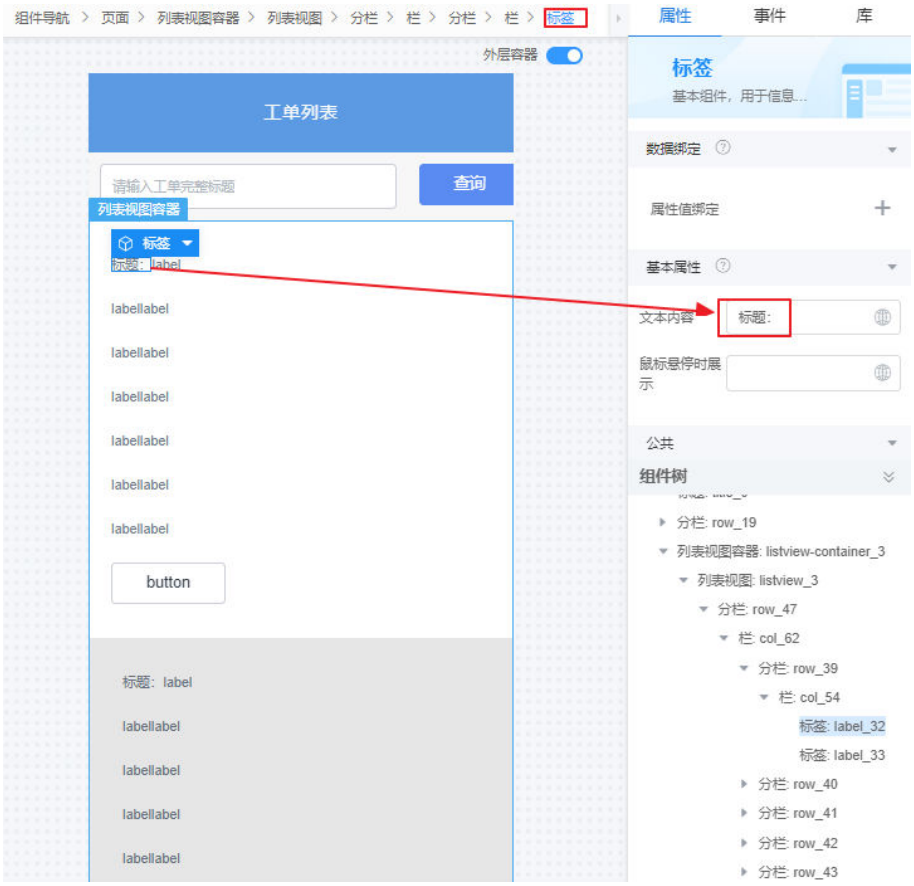
图 2-223 拖入列表视图

2. 在“列表视图容器”**下层的“列表视图”**中，拖入1个“分栏”，在右侧修改为1个栏（Phone），并设置分栏内边距为“10px”。

图 2-224 修改分栏为 1 栏及内边距

3. 在“分栏”中，再拖入1个“分栏”，并参考上图修改为1个栏（Phone），然后单击分栏的⊕7次，新增7个分栏，新增后，变为8个分栏。
4. 向第1~7分栏中，分别拖拽2个“标签”组件（并排），然后在第8分栏中，拖入一个“按钮”组件。
5. 修改第1分栏中的第1个标签的“文本内容”修改为“标题：”。

图 2-225 修改第 1 个标签标题



6. 重复上一步，修改第2~7分栏中的第1个标签的“文本内容”为“单号：”、“状态：”、“时间：”和“优先级：”。

表 2-28 标签的文本内容

所在分栏	第一栏	第二栏
1	标题：	保持默认（label）
2	单号：	保持默认（label）
3	状态：	保持默认（label）
4	要求到达时间：	保持默认（label）
5	要求解决时间：	保持默认（label）
6	现场故障说明：	保持默认（label）
7	优先级：	保持默认（label）

说明

正常情况下，第二栏中的内容是动态获取的，在**表2-30**中，设置的第二栏中的“文本内容”，仅会在未获取相应工单信息时显示，您也可以根据实际需要自定义“标题”组件默认显示内容。

7. 在“样式”属性下，自定义修改各个标签组件的样式，例如加粗文本。

图 2-226 自定义修改样式

8. 在第8行内，选中“按钮”组件，设置按钮“显示名称”为“处理”、“类型”为“主要按钮”，您也可以在属性中为按钮添加图标，修改按钮背景色为“#4a90e2”。

   组装完成效果，如图2-227所示。

图 2-227 组装工单信息展示区域

步骤8 单击界面上方的，保存页面。

----结束

定义模型并绑定页面组件

页面模型负责与页面组件交互，获取显示工单需要的数据。结合页面展示需求，可知需要创建如下模型：

表 2-29 模型分析

模型名称	作用	详细定义
workOrderList	保存工单的基本信息。	自定义模型，包含的字段节点如下，这些节点与工单对象的字段名称相同，**加粗斜体内容**请替换为实际命名空间前缀： • ***HW***__title__CST：工单标题，字段类型Text。 • ***HW***__workOrderId__CST：工单ID，字段类型Text。 • ***HW***__priority__CST：工单优先级，字段类型Text。 • id：记录ID，字段类型Text。 • ***HW***__recoveryTime__CST：要求解决时间，字段类型Text。 • ***HW***__faultPhenomenon__CST：现场故障说明，字段类型Text。 • ***HW***__status__CST：工单状态，字段类型Text。 • ***HW***__instanceId__CST：工单绑定的BPM实例的id，字段类型Text。 • ***HW***__arriveTime__CST：要求到达时间，字段类型Text。 • isDeal：判断工单状态字段，字段类型Bool。
queryCondition	查询条件。	自定义模型，包含字段节点：title，字段类型Text，与页面上的查询条件对应。
queryWorkOrder	查询工单。	服务模型，绑定公共接口“ queryWorkOrder”，调用查询工单逻辑。
total	保存满足查询条件的记录数，作为Scroll组件滚动显示时的累计记录数。	自定义模型，包含的字段节点如下： • currentTotal：当前页显示的数量，字段类型Text。 • actualTotal：实际总数量，字段类型Text。

步骤1 定义模型“workOrderList”。

1. 单击“模型视图”，切换到模型视图。
2. 在模型视图中，单击“新增模型”。
3. 添加自定义模型，模型名称“workOrderList”，单击“下一步”。
4. 单击“新增节点”，依次增加表2-29中列出的字段名称，单击“下一步”，再单击“确定”。
5. 单击页面上方的，保存模型。

步骤2 定义模型“queryCondition”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加自定义模型，模型名称“queryCondition”，单击“下一步”。
3. 单击“新增节点”，依次增加表2-29中列出的字段名称，单击“下一步”，再单击“确定”。
4. 单击页面上方的，保存模型。

步骤3 定义模型“total”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加自定义模型，模型名称“total”，单击“下一步”。
3. 单击“新增节点”，依次增加表2-29中列出的字段名称，单击“下一步”，再单击“确定”。
4. 单击页面上方的，保存模型。

步骤4 定义模型“queryWorkOrder”。

1. 在“模型视图”中，单击“新增模型”。
2. 添加服务模型，模型名称“queryWorkOrder”，单击“下一步”，
3. “服务类型”选择“公共接口”，“选择项目”为“设备维修管理系统”，“搜索”中的接口选择“queryWorkOrder”，单击“下一步”，再单击“确定”。
4. 单击页面上方的，保存模型。

步骤5 绑定模型“workOrderList”。

1. 单击“设计视图”，从“模型视图”切换到“设计视图”。
2. 选中“列表视图容器”标签中的**“列表视图”**标签（不要选择列表视图容器），为其绑定“workOrderList”。

图 2-228 绑定数据模型

3. 如图2-229所示，选中“标题：”**后的标签组件**label，在右侧“属性”页签，单击“+”，增加属性值绑定，即设置“属性”为“文本内容”，“模型字段”为“workOrderList”下的“***HW***__title__CST”。

图 2-229 为标签绑定模型

4. 依次为其他标签添加属性值绑定，具体绑定的字段如表2-30所示。

表 2-30 标签的属性值绑定

所在分栏	第一栏	第二栏	第二栏标签绑定的字段
1	标题:	label	workOrderList.***HW***__title__CST
2	单号:	label	workOrderList.***HW***__workOrderId__CST
3	状态:	label	workOrderList.***HW***__status__CST
4	要求到达时间:	label	workOrderList.***HW***__arriveTime__CST
5	要求解决时间:	label	workOrderList.***HW***__recoveryTime__CST
6	现场故障说明:	label	workOrderList.***HW***__faultPhenomenon__CST
7	优先级:	label	workOrderList.***HW***__priority__CST

5. 为“处理”按钮添加属性值绑定，“属性”设置为“禁用”，绑定“workOrderList”的“isDeal”字段，绑定后为“workOrderList.isDeal”。

图 2-230 添加属性值绑定

6. 单击页面上方的，保存设置。

步骤6 绑定模型“queryCondition”。

1. 选中“请输入工单完整标题”输入框，为其进行值绑定，绑定字段为“queryCondition.title”。

图 2-231 绑定工单标题

2. 单击页面上方的，保存设置。

----结束

定义页面中的事件代码

通过定义页面的on-load事件，可以实现打开页面即自动展示当前工程师的待处理工单。

步骤1 定义页面事件代码。

1. 在“设计视图”中，选中最外层的“页面”。

2. 在右侧“事件”页签中，单击“加载”后的“+”。
3. 在添加事件弹窗中，输入如下脚本代码。

```
$model.ref('queryWorkOrder').setData({inputParam: {"isFME": true}});
$model.ref('queryWorkOrder').run().then(function (data) {
    if (data && data.workOrderList) {
        //将查询到的数据赋值给页面模型来展示
        $model.ref('workOrderList').setData(data.workOrderList);
    }
}).catch(function (error) {
    console.log('error is', error);
});
```

4. 单击“创建”，退出事件编排窗口。

----结束

实现“按标题查询工单”

通过定义“查询”按钮的“点击”事件，可以实现根据工单标题查询工单的能力。

步骤1 在“设计视图”中，选中“查询”按钮。

步骤2 在右侧“事件”页签中，单击“点击”后的“+”。

步骤3 在添加事件弹窗中，输入如下脚本代码。

```
let queryTitle = $model.ref('queryCondition').getData().title;
$model.ref('queryWorkOrder').setData({ inputParam: { "title": queryTitle, "isFME": true } });
$model.ref('queryWorkOrder').run().then(function (data) {
    if (data && data.workOrderList) {
        //将查询到的数据赋值给页面模型来展示
        $model.ref('workOrderList').setData(data.workOrderList);
    }
}).catch(function (error) {
    console.log('error is', error);
});
```

步骤4 单击“创建”，退出事件编排窗口。

----结束

实现“处理工单”

通过定义“处理”按钮的“点击”事件，可以实现接单、处理工单的能力。

步骤1 在“设计视图”中，选中“处理”按钮。

步骤2 在右侧“事件”页签中，单击“点击”后的“+”。

步骤3 在添加事件弹窗中，输入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的页面名。

```
//获取当前组件（即button）
let _component = context.$component.current;
// 配置页面的bpm参数
context.$page.params["bp.name"] = "HW__WorkOrderBpm";
//获取当前行id
let id = _component.$attrs.row.id;
let taskId = "";
```

```
let instanceId = "";
//遍历获取当前行内容
let row = {};
let workorders = $model.ref('workOrderList').getData();
workorders.forEach(function (wo, idx) {
    if (wo.id == id) {
        row = wo;
        instanceId = wo.HW__instanceId__CST;
    }
});
//打开处理弹框
context.$dialog.popup({
    title: '处理工单',
    page: 'HW__workOrderProcess',
    footerHide: false,
    width: 20,
    height: 180,
    okText: '提交',
    cancelText: '取消',
    params: { id: id, row: row },
    onCancel: function () {
    },
    onOk: function () {
        let statusInfo = { "statusInfo": $model.ref("statusInfo").getData() };
        let taskData = {
            "action": "complete",
            "variables": statusInfo
        };

        // 获取csrf token
        context.$utils.getCSRFToken().then(function (token) {
            let url = '/u-route/baas/bp/v2.0/query/tasks?flag=activeTask&rootID=' + instanceId;
            fetch(url, {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'CSRF-Token': token
                }
            }).then(function (resp) {
                resp.json().then(function (data) {
                    taskId = data.result.Recs[0].id;
                    let url2 = '/u-route/baas/bp/v2.0/runtime/tasks/' + taskId;
                    fetch(url2, {
                        method: 'PUT',
                        headers: {
                            'Content-Type': 'application/json'
                        },
                        body: JSON.stringify(taskData)
                    }).then(function (resp) {
                        context.$message.success('处理成功');
                        $model.ref('queryWorkOrder').setData({ inputParam: {"isFME": true} });
                        $model.ref('queryWorkOrder').run().then(function (data) {
                            if (data && data.workOrderList) {
                                //将查询到的数据赋值给页面模型来展示
                                $model.ref('workOrderList').setData(data.workOrderList);
                            }
                        }).catch(function (error) {
                            console.log('error is', error);
                        });
                    }).catch(function (error) {
                        console.log('error is', error);
                    });
                });
            }).catch(function (error) {
                console.log('error is', error);
            });
        });
    }
});
```

步骤4 单击“创建”，退出事件编排窗口。

----结束

验证

通过**PC端**预览页面的方式，访问“待处理工单”页面进行测试。

说明

请在PC端预览或测试“待处理工单”标准页面，当前版本不建议在手机端预览页面。

步骤1 打开“待处理工单”标准页面。

步骤2 单击界面上方的  ，进入预览页面。

步骤3 查看页面的展示效果。

图 2-232 工单列表页面

如果当前租户开发者名下没有工单，所以当前的预览效果只能看到页面框架，没有工单。

图 2-233 无工单信息

----结束

2.7.6 开发“管理员管理工单”功能

2.7.6.1 创建“删除工单”脚本及公共接口

管理员的“工单管理”页面与“工单列表（派单）”页面相比，功能大致相同，仅多了一个创建工单功能、操作列中的删除按钮，因此需要多创建一个实现删除工单功能的后台逻辑。

创建脚本

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“WorkOrder”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“deleteWorkOrder”，单击“添加”。

当编辑已有脚本时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 在代码编辑器中插入如下脚本代码。

须知

脚本中**红色内容**请替换为实际的对象名、字段名。

```
//本脚本用于删除工单
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
//定义入参结构，入参包含1个参数：workOrder对象，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    id: string;
}
//定义出参结构，出参包含1个参数，workOrder的记录id
```

```
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
}
//使用数据对象HW__WorkOrder__CST
@useObject(['HW__WorkOrder__CST'])
@action.object({ type: "method" })
export class DeleteWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public deleteWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let id = input.id;
            let s = db.object('HW__WorkOrder__CST');    //获取HW__WorkOrder__CST这个Object的操作实例
            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "id",
                    "operator": "eq",
                    "value": id
                }]
            };
            let isDeleted = s.deleteByCondition(condition);
            if (isDeleted) {
                out.id = id;
            } else {
                error.name = "WOERROR";
                error.message = "删除工单失败！";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

步骤5 单击编辑器上方的  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的  ，执行脚本。
2. 在界面底部，输入以下报文作为“输入参数”，然后单击测试窗口右上角  执行图标。“*ceHg000000e0glLbDQ2K*”则是派单员操作的当前工单记录的工单id，您可以在**派单功能**脚本的测试结果中获取一个id值。

   ```
   {
       "id":"ceHg000000e0glLbDQ2K"
   }
   ```

   执行成功，会在“输出”页签返回查询结果。

步骤7 测试成功，单击编辑器上方的  ，启用发布脚本。

----结束

创建公共接口

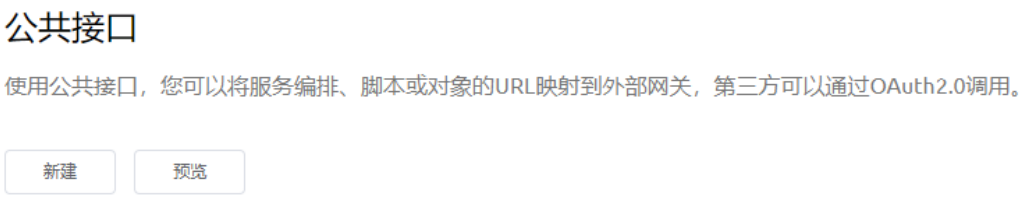
参考之前创建公共接口的步骤，创建“删除工单”脚本对应的公共接口。

步骤1 在设计视图下，单击下方“服务”，进入公共接口创建页面。

图 2-234 创建公共接口入口

步骤2 在公共接口中，单击“新建”。

图 2-235 公共接口创建



步骤3 创建“删除工单”脚本对应的公共接口，详细接口信息如表2-31所示。

表 2-31 公共接口

设置操作	版本	URL	方法	类型	资源
deleteWorkOrder	1.0.0	/deleteWorkOrder	DELETE	脚本	***HW***__deleteWorkOrder

----结束

2.7.6.2 组装“工单管理”页面

管理员的“工单管理”页面与“工单列表（派单员）”页面相比，大致功能一致，仅多了一个创建工单功能，和操作列中的删除按钮。因为两个页面很相似，所以为了快速创建，将通过“工单列表（派单员）”页面另存一个页面，然后再改造这个另存后的页面成为“管理员管理工单”页面。

“工单管理”页面是在“工单列表（派单员）”基础上创建的，因此该页面创建完成后，就已经具备了“查询工单”、“派发工单”的功能，而在本节的开发步骤中，主要为该页面添加“创建工单”、“删除工单”功能即可。其中，“创建工单”功能与“工单列表（客服人员）”页面中的创建方法基本一致。

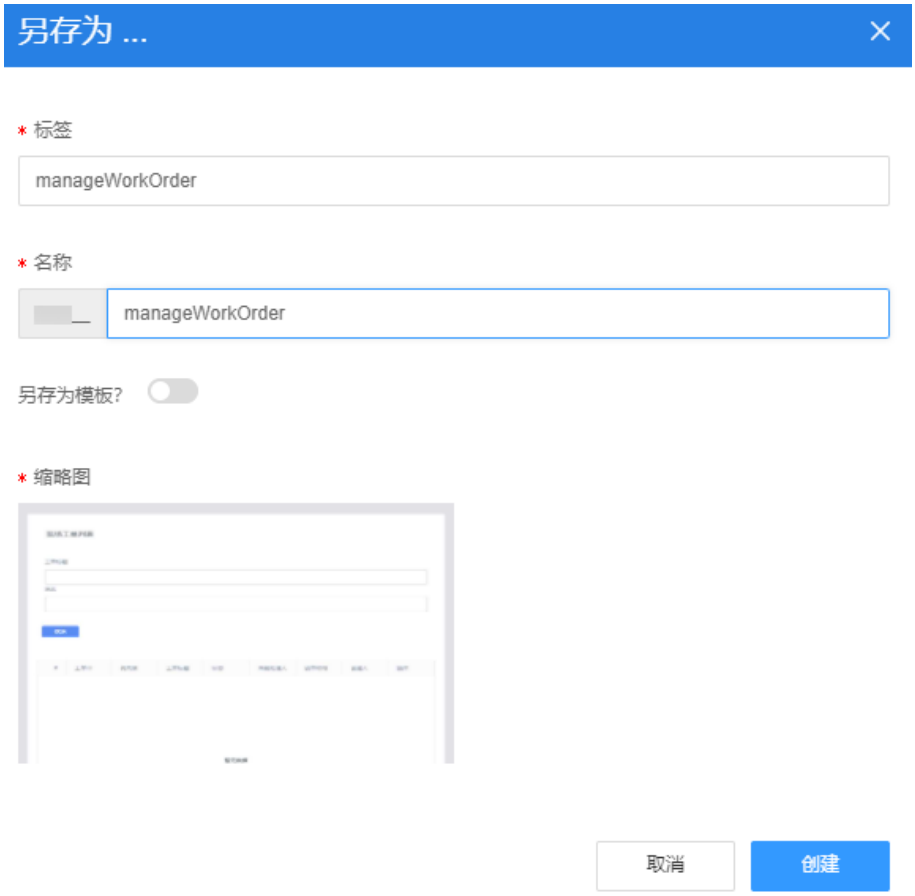
操作步骤

步骤1 在经典版开发环境“首页 > 我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 使用“工单列表（派单员）”页面创建模板。

1. 打开“***HW***_dispatchWorkOrder”，单击锁定页面。
2. 单击页面上方的，保存模板。
3. 如图2-236所示，在弹出对话框中设置页面标签和名称为“manageWorkOrder”，单击“创建”。

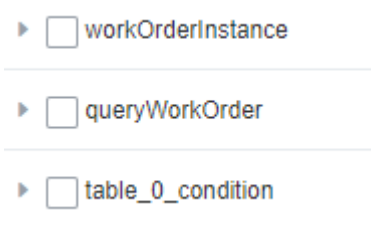
图 2-236 保存页面模板



步骤3 检查及创建页面模型。

1. 一般情况下，另存方式创建的页面，会保留原页面中的页面模型，单击“模型视图”，查看当前页面是否存在以下模型。

图 2-237 页面原有模型



2. 创建服务模型“deleteWorkOrder”。该模型用于绑定删除工单公共接口。
   a. 在“模型视图”，单击“新增模型”。
   b. 添加服务模型，模型名称“deleteWorkOrder”，单击“下一步”，
   c. “服务类型”选择“公共接口”，“选择项目”为“设备维修管理系统”，“搜索”中的接口选择“deleteWorkOrder”，单击“下一步”，再单击“确定”。
   d. 单击页面上方的，保存模型。

步骤4 添加“创建工单”按钮。

1. 单击“设计视图”，从“模型视图”切换到“设计视图”。
2. 在页面中，选中“表格”，单击右侧“属性 > 表格区块”中“工具栏”后的“添加”按钮。

图 2-238 增加查询条件区域

3. 删除工具栏中多余的按钮，只保留“新增行”。
4. 修改“新增行”的“显示名称”为“创建工单”。
5. 单击页面上方的，保存设置。

步骤5 为操作列增加“删除”功能。

1. 选中“表格”，在右侧“属性”页签的“表格列”中，单击上图中“操作”列后的 。

图 2-239 编辑操作列

2. 在“属性配置”对话框中，单击“添加操作按钮”，新增后按钮在“派单”按钮下方，然后对新增按钮进行以下设置。

   a. 向下拖拽拖动条，找到新增的按钮区域，设置按钮“标签”为“删除”，“图标”选择“delete”。

图 2-240 为操作列增加派单按钮

b. 单击图2-240中“删除”按钮的，再单击“动作列表”的“+”，进入事件编排中，在事件编排页面，添加如下事件代码，然后单击“创建”。

须知

脚本中**红色内容**请替换为实际的页面名、组件名。

```
let _component = context.$component.current
let currentRow = _component.$attrs.row;
let id = currentRow.id;
let _model = $model.ref("workOrderInstance");
context.$dialog.confirm(
   {
      title: '确认',
      content: '确认删除该工单吗？',
      okText: '确定',
      cancelText: '取消',
      onOk: () => {
         $model.ref('deleteWorkOrder').setData({ inputParam: { "id": id } });
         $model.ref('deleteWorkOrder').run().then(function (data) {
            let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
            let queryData = {
               "start": (pageInfo.curPage - 1) * pageInfo.pageSize,
               "limit": pageInfo.pageSize
```

```
            };
            $model.ref('queryWorkOrder').setData({ inputParam: queryData });
            $model.ref('queryWorkOrder').run().then(function (data) {
                $model.ref('workOrderInstance').setData(data.workOrderList);
                pageInfo.total = parseInt(data.total);
                $model.ref('table_0_condition').setData({ "pageInfo": pageInfo });
            }).catch(function (error) {
                console.log('error is', error);
            });
        });
    },
    onCancel: () => { }
  }
);
```

c. 单击“确定”，关闭“属性配置”对话框。

d. 单击页面上方的，保存设置。

步骤6 **检查**页面加载事件是否跟随页面复制过来。

1. 选中最外层的“页面”，在右侧“事件”页签中，单击“自定义JS代码”后的。
2. 查看自定义动作中是否已有如下脚本代码，如果没有，请输入。

```
let pageInfo = $model.ref('table_0_condition').getData().pageInfo;
let queryData = {
    "start": (pageInfo.curPage - 1) * pageInfo.pageSize,
    "limit": pageInfo.pageSize
};
$model.ref('queryWorkOrder').setData({ inputParam: queryData });
$model.ref('queryWorkOrder').run().then(function (data) {
    $model.ref('workOrderInstance').setData(data.workOrderList);
    pageInfo.total = parseInt(data.total);
    $model.ref('table_0_condition').setData({ "pageInfo": pageInfo });
}).catch(function (error) {
    console.log('error is', error);
});
```

3. 单击“创建”，关闭事件编排器，返回到页面。

步骤7 添加“创建工单”按钮跳转事件，跳转事件功能是将页面跳转到“生成工单”页面。

1. 选中“创建工单”按钮，在右侧“事件”页签中，单击“新增行”后的，进入事件编排页面。
2. 在“自定义动作”中，输入以下事件代码，加粗斜体内容请替换为实际命名空间前缀。

```
//打开创建工单页面，需要根据实际页面名称修改
context.$page.load('/besBaas/page#/HW__createWorkOrder')
```

3. 单击“创建”，关闭事件编排器，返回到页面。
4. 单击页面上方的，保存页面。

----结束

2.7.7 定义工单流转 BPM

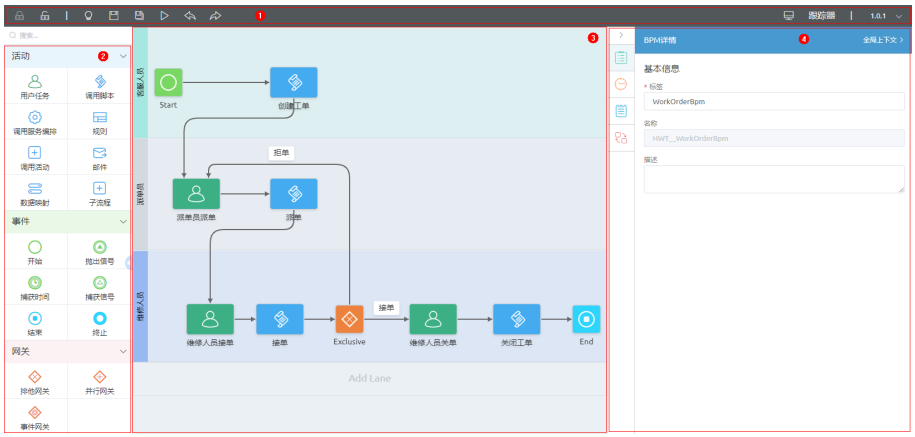
2.7.7.1 背景与原理（BPM）

工单管理模块中的工单场景业务编排是通过AstroZero的流程编排BPM（Business Process Management）功能实现的，通过在前端页面调用BPM完成工单流转，即客服人员创单，派单员派发工单，维修工程师处理工单的全过程。

开发BPM即是对前端页面及后端逻辑（脚本、服务编排等）进行编排的过程。AstroZero提供的BPM作为商业流的配置工具，可以通过模板化、图形化实现对商业流业务流程的编排和执行。

了解 BPM 设计界面

图 2-241 BPM 设计界面



整体编辑器页面由上方按钮区域、左侧面板图元区域、中间画布工作区域、右侧属性配置区域四部分组成。

表 2-32 BPM 设计界面说明

编号区域名称	功能说明
1	功能按钮区域，包括锁定、解锁、启用（或者禁用）、保存、另存为新版本或者新BPM、运行、操作回退、撤销回退、启用流跟踪器以及切换版本的操作。支持快捷键操作，即可脱离鼠标直接用键盘操作。
2	BPM的组成图元，一个BPM业务流程由以下几个部分组成： • 事件图元（Events）：用来表明BPM的生命周期中发生的事件，例如开始、捕获信号等。 • 网关图元（Gateways）：网关用来控制流程的执行流向，可理解为决策、判断。 • 活动图元（Activities）：是BPM的核心图元，可理解为节点或者步骤，例如调用脚本、用户需要做的任务。
3	BPM设计操作区域。在该区域可对BPM进行具体流程设计、组件放置。 • 不同色块的表示不同的泳道，BPM由一个或多个泳道组成，泳道中包括了实现不同功能逻辑的图元。 • 选中泳道或者某个图元，可以在右侧属性配置区域进行各种配置操作。

编号区域名称	功能说明
4	整个BPM、泳道或者图元属性设置区域。 ● 当选择BPM中图元时，右侧配置区域为该图元的属性设置区域。 ● 当选择BPM中空白区域时，右侧配置区域为该BPM的设置区域。 ● 当选择泳道左侧标签时，右侧配置区域为该泳道的属性设置区域。

BPM 能力

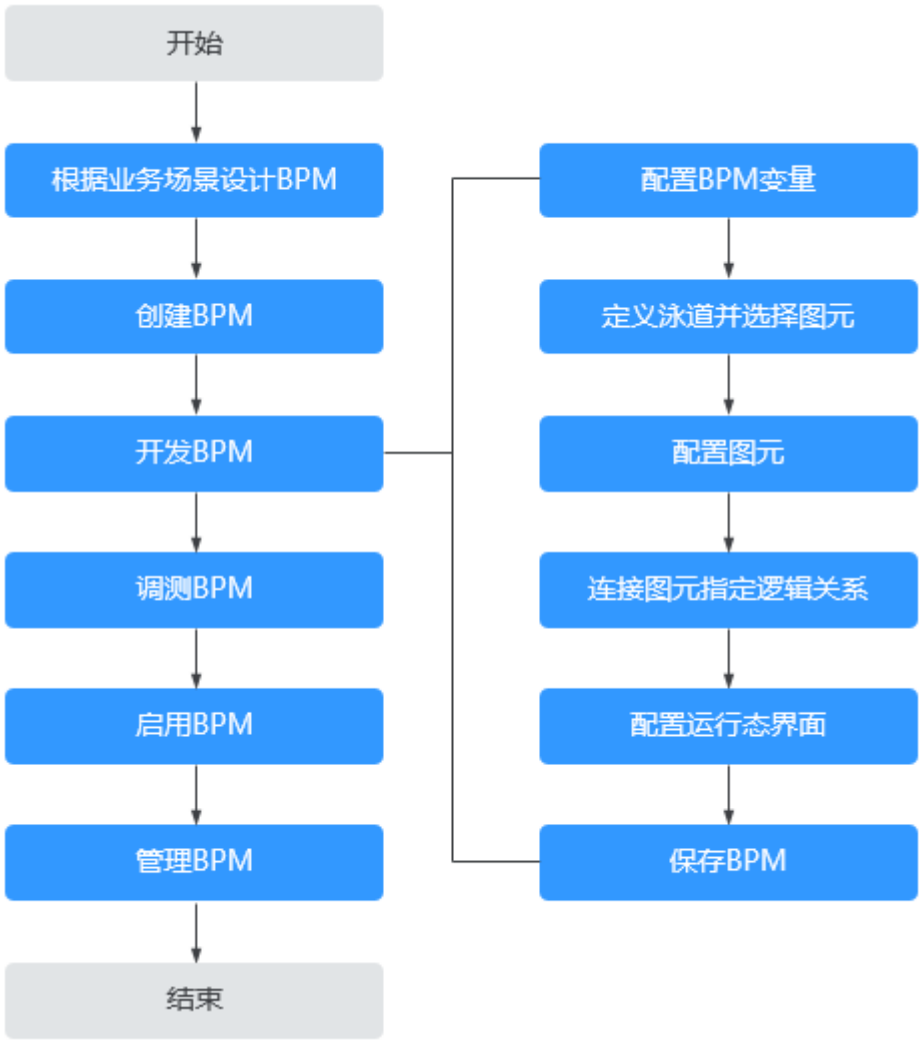
BPM可以提供如下能力，本示例中仅使用到前三种能力：

- 端到端页面流编排
  将用户交互的前端页面与各种任务之间进行编排，形成一个完整的商业流。
- 支持长流程
  步骤之间可以是立即执行，也可以是小时、天、甚至更长时间的间隔，支持SLA期限管理。
- 跨人员的工作流
  支持每一步由不同的用户、组串行或并行处理。
- BPM内部进行调用
  BPM可以作为子BPM被其他BPM进行内部调用。在总的BPM中使用“调用活动”元素，可嵌套使用子BPM。

BPM 开发流程

BPM开发的一般流程如图2-242所示。

图 2-242 BPM 开发配置大致流程



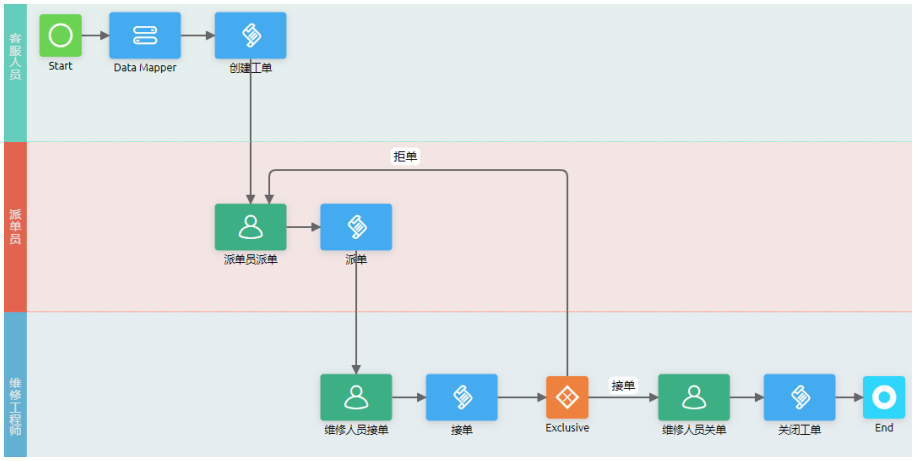
更多BPM详细介绍，请参见**Astro轻应用产品文档**中相关描述。

2.7.7.2 创建并开发 BPM

工单管理开发中，工单流转BPM设计大致如下。

1. 根据业务角色不同，使用不同的工作队列，创建客服人员、派单员以及维修人员3个泳道。

图 2-243 工单 BPM 设计界面



通过在不同的泳道中，拖拽功能图元，然后通过配置图元属性，以及需要调用关键脚本逻辑，将前端页面中获取到的数据传回到后端的工单对象中，以完成工单流转的全过程。图元之间调用关系及工单流转过程，如表2-33所示。

表 2-33 图元调用关系及工单流程过程

序号	图元名称	图元类型	图元作用	下一步流转
1	Start	开始	用户在创建工单页面，填写工单信息并单击“提交”按钮后，触发该图元，并将工单信息传入。	携带工单信息流转到“Data Mapper”图元。
2	Data Mapper	数据映射	将当前BPM的ID赋值给工单对象中BPM的ID字段。	携带工单信息、BPM的信息流转到“创建工单”图元。
3	创建工单	调用脚本	调用“创建工单”脚本，根据传入的工单信息，完成工单创建。	工单创建完毕后，流转“派单员派单”图元。
4	派单员派单	用户任务	将创建成功的工单，流转到派单员的派单页面（派单员队列中的用户均可进行派单）。	单击“派单”按钮，选择需要指定的“维修人员”后，将派单信息流转到“派单”图元。
5	派单	调用脚本	调用“派发工单”脚本，根据传入的派单信息进行工单信息（当前处理人/状态）的修改。	“派发工单”脚本执行后，会返回维修人员的用户Id，工单将流转到“维修人员”图元。

序号	图元名称	图元类型	图元作用	下一步流转
6	维修人员接单	用户任务	将派发成功的工单流转到指定的维修人员的工单处理界面。	单击“处理”按钮，进行工单的处理，可选择“接单”或“拒单”，并将处理信息流转到“接单”图元。
7	接单	调用脚本	调用“处理工单”脚本，根据传入的处理信息进行工单信息（当前处理人/状态）的修改。	“处理工单”脚本执行后返回处理结果，并将处理结果流转到“Exclusive”排他网关图元，进行下一步判断。
8	Exclusive	排他网关	根据脚本返回的处理结果进行判断，执行不同的流程。	• 如果脚本返回结果为“接单”，将流转到“维修人员关单”图元。 • 如果脚本返回结果为“拒单”，将重新流转回“派单员派单”图元，重新进行派单。
9	维修人员关单	用户任务	将接单成功的工单流转到接单的维修人员的工单处理界面。	对于处理中状态的工单，单击“处理”按钮，进行工单的处理，即进行关闭操作，将处理信息流转到“关闭工单”图元。
10	关闭工单	调用脚本	调用处理工单脚本，根据传入的处理信息，将工单状态修改为关闭。	关闭工单成功后，将流转到结束图元。
11	End	结束	整个流程执行完毕。	-

操作步骤

创建流程大致是先创建3个泳道，再拖拽泳道图元，然后配置相关图元，保存并启用。

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

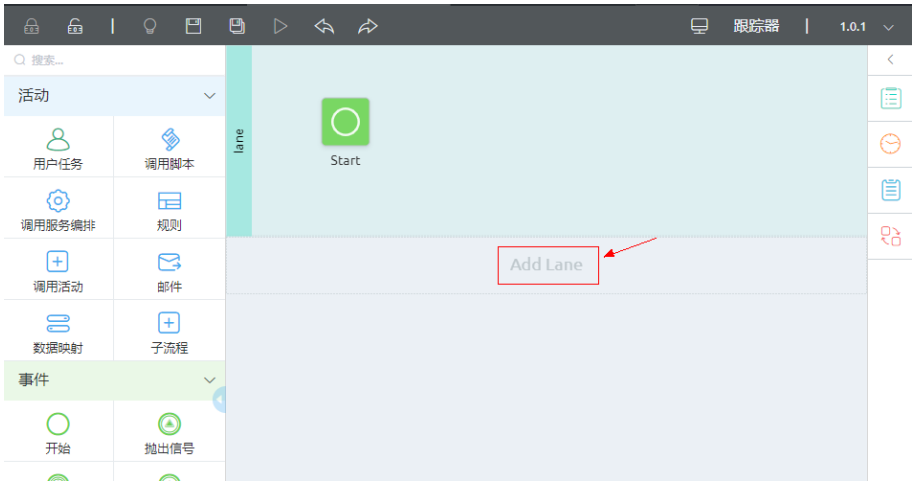
步骤2 在“WorkOrder”目录中，将鼠标放在“Bpm”上，单击界面上出现的“+”，在弹出菜单中选择“BPM”。

步骤3 在弹窗中，设置“标签”、“名称”为“WorkOrderBpm”，单击“添加”。

图 2-244 添加 BPM

步骤4 在设计区域，单击2次“AddLane”，为BPM添加2个角色泳道。

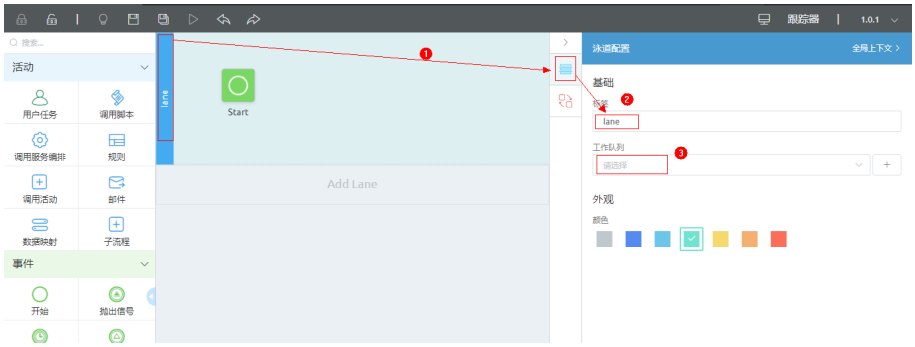
图 2-245 添加泳道



步骤5 修改泳道基础属性。

1. 选中“Lane”，在右侧属性配置区域，设置泳道标签为“客服人员”，“工作队列”设置为**创建工作队列**中创建的“客服人员”。

图 2-246 修改客服人员泳道基本信息



2. 参考上一步，分别设置其他两个泳道的“标签”及“工作队列”。

表 2-34 泳道标签及工作队列

泳道初始标签	泳道修改后标签	工作队列
Lane（上一步已配置）	客服人员	客服人员
lane1	派单员	派单员
lane2	维修人员	维修人员

步骤6 创建BPM中用的变量及对象变量。

1. 在右侧属性面板区域，单击  ，展开全局上下文。

图 2-247 展开全局上下文

2. 单击“变量”后的  图标4次，分别创建以下变量，单击变量后的  ，可以修改字段类型，创建完成后，如图2-248所示。

表 2-35 BPM 变量

变量名	字段类型
id	文本
status	文本
assignedFme	文本
transInfo	任意

图 2-248 创建变量

3. 单击“对象变量”的，在弹窗中设置“名称”为“workOrderData”，“对象”设置为“***HW***__WorkOrder__CST”（工单对象），加粗斜体内容请以实际命名空间前缀为准，然后单击“保存”。

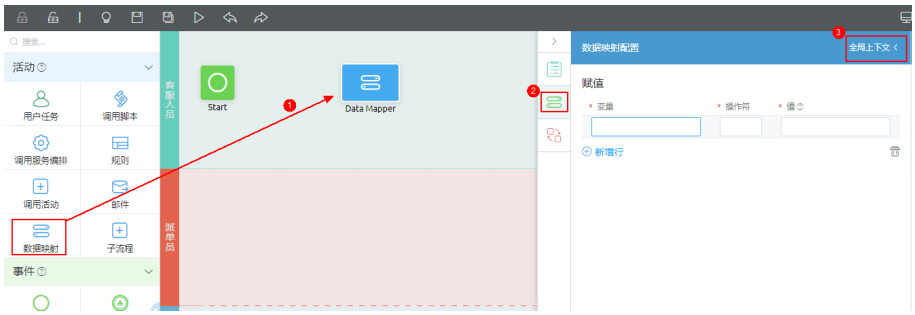
图 2-249 新增对象变量

4. 再次单击“对象变量”的，在弹窗中设置“名称”为“statusInfo”，“对象”设置为“***HW***__WorkOrder__CST”（工单对象），然后单击“保存”。

步骤7 拖拽并配置“客服人员”泳道。

1. 从左侧“活动”图元区域，拖拽一个“数据映射”图元到“客服人员”泳道中。
2. 选中“Data Mapper”图元，然后在右侧导航菜单上单击，进入数据映射配置，再单击，展开全局上下文。

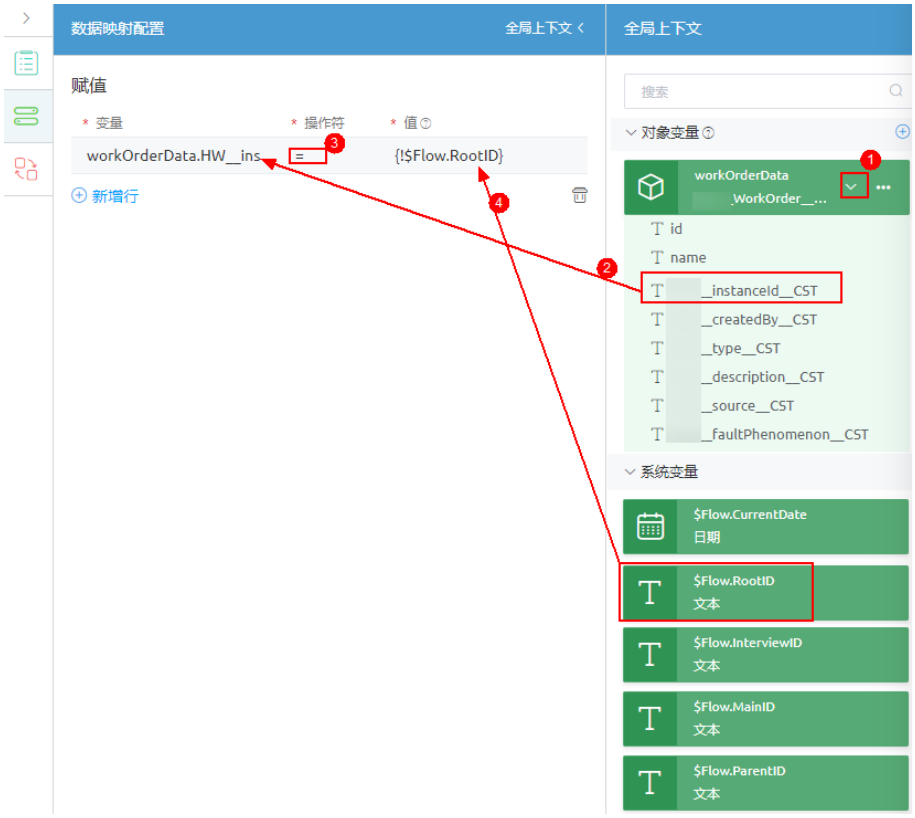
图 2-250 拖拽并设置“数据映射”图元



3. 分别从全局上下文的“对象变量”、“系统变量”中拖拽参数到数据映射配置下：

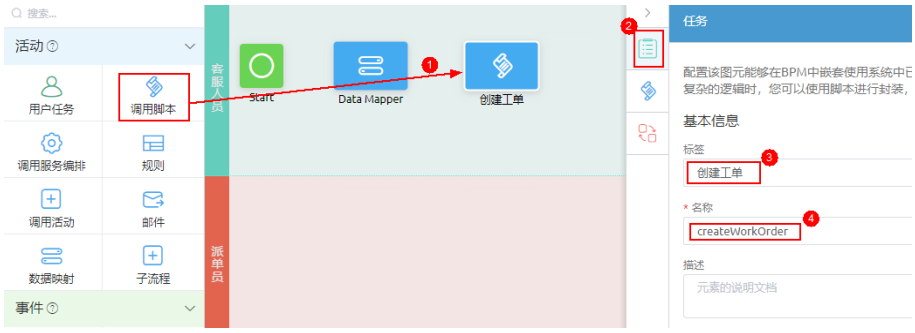
   拖拽对象变量“workOrderData”下的“***HW***__instanceId__CST”字段到“变量”下，“操作符”为“=”，拖拽“系统变量”下的“{!$Flow.RootID}”作为“值”。

图 2-251 拖拽赋值变量



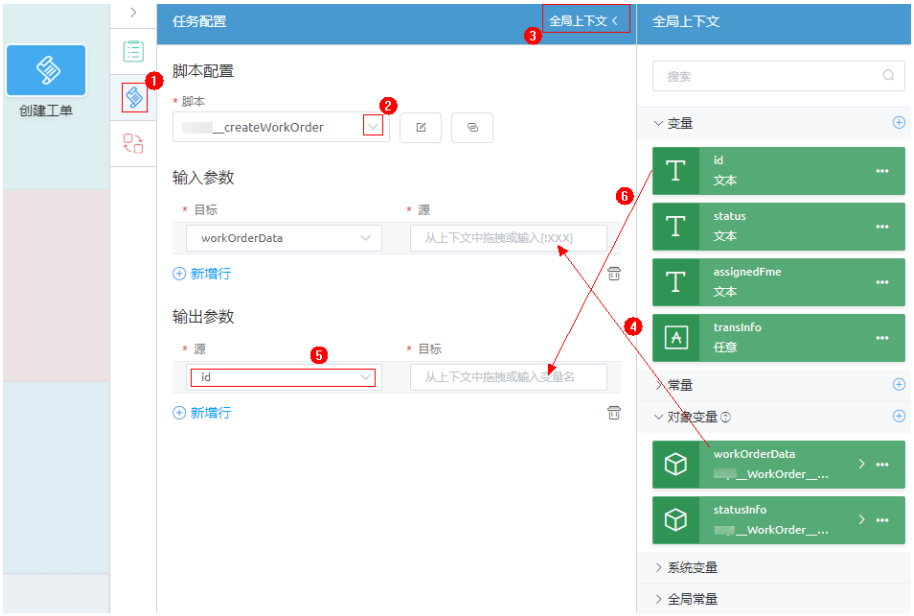
4. 从左侧“活动”图元区域，拖拽一个“调用脚本”图元到“客服人员”泳道中，并设置“标签”为“创建工单”、“名称”为“createWorkOrder”。

图 2-252 设置创建工单图元



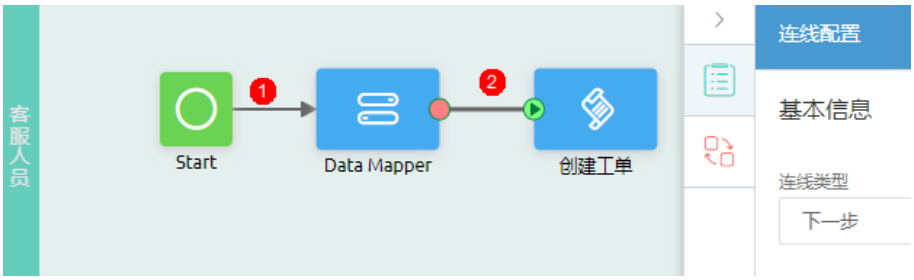
5. 选中“创建工单”图元，单击，再单击“全局上下文”，配置“创建工单”图元绑定的脚本以及输入输出参数。
   a. 在“脚本配置”下选择“生成工单”脚本“***HW***__createWorkOrder”。
   b. 从全局上下文中拖拽对象变量“workOrderData”到“输入参数”的“源”输入框中，作为输入参数的值。
   c. 设置“输出参数”为“id”，并拖拽变量“id”作为“目标”输入框中的值。

图 2-253 配置“创建工单”图元



6. 从“Start”图元中拖拽一条连线到“Data Mapper”图元，然后从“Data Mapper”图元拖拽一条连线到“创建工单”图元，默认“连线类型”为“下一步”。

图 2-254 拖拽连线



步骤8 拖拽并配置“派单员”泳道。

1. 从左侧“活动”图元区域，依次拖拽一个“用户任务”、“调用脚本”图标到“派单员”泳道中。
2. 设置“用户任务”图元的“标签”为“派单员派单”、“名称”为“dispatchWorkOrderList”。
3. 设置“调用脚本”图元的“标签”为“派单”、“名称”为“dispatch”。
4. 选中“派单员派单”图元，单击，设置“任务标题”、“任务描述”为“派单员派单”，“渲染类型”为“标准页面”，“页面”设置为“***HW***__dispatchWorkOrder”。

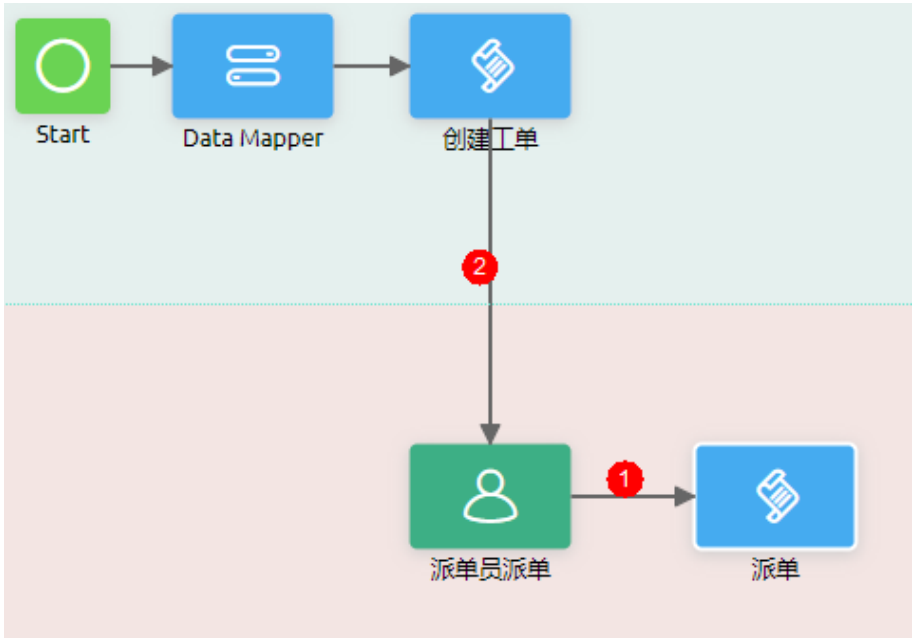
图 2-255 用户任务配置

5. 选中“派单”图元，单击，再单击“全局上下文”，配置“派单”图元绑定的脚本以及输入输出参数。
   a. 在“脚本配置”下选择“生成工单”脚本“***HW***__dispatchWorkOrder”。
   b. 从全局上下文中，拖拽变量“transInfo”到“输入参数”的“源”输入框中，作为输入参数的值。
   c. 在“输出参数”下，单击“新增行”，设置“输入参数”为“id”、“assignedFme”，并从全局上下文中，**拖拽**变量“id”、“assignedFme”作为“目标”输入框中的值。

图 2-256 配置“派单”图元

d. 从“派单员派单”图元中拖拽一条连线到“派单”图元，然后再从“创建工单”图元中拖拽一条连线到“派单员派单”图元，默认“连线类型”均为“下一步”。

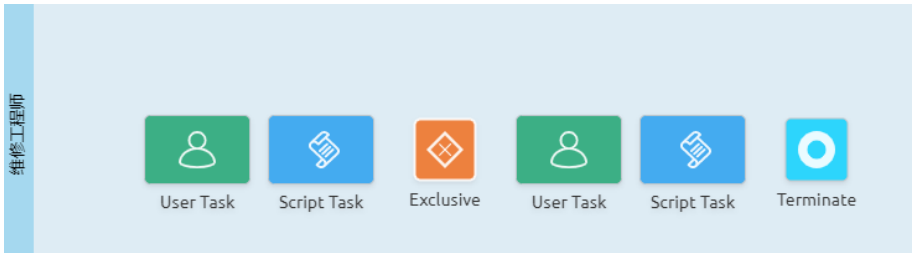
图 2-257 拖拽连线



步骤9 拖拽并配置“维修人员”泳道。

1. 从左侧“活动”图标区域，拖拽2个“用户任务”、2个“调用脚本”、1个“排他网关”以及“终止”图元到“维修人员”泳道中，然后调整图元顺序。

图 2-258 拖拽并调整图元顺序

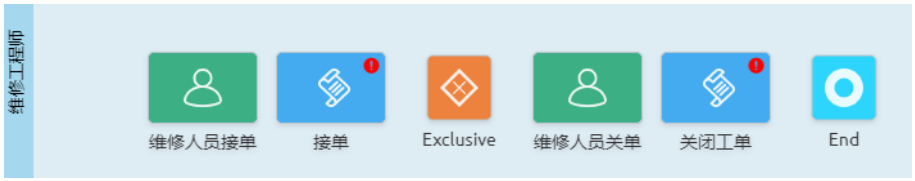


2. 从左到右，分别设置2个“用户任务”、2个“调用脚本”图元的基本属性，具体如**表2-36**所示，设置完成后如**图2-259**所示。

表 2-36 图元基本属性设置

图元（从左向右）	标签	名称
用户任务1	维修人员接单	takeWorkOrder
调用脚本1	接单	takeWorkOrder1
用户任务2	维修人员关单	dealWorkOrder
调用脚本2	关闭工单	closeWorkOrder
终止	End	end

图 2-259 图元基本属性设置



3. 配置“维修人员接单”图元。

选中“维修人员接单”图元，单击  ，并展开全局上下文，配置“维修人员接单”图元，详细配置如**表2-37**所示。

图 2-260 用户任务配置

表 2-37 用户任务配置

参数项	值
任务标题	维修人员接单
任务描述	维修人员任务列表
渲染类型	标准页面
页面	***HW***__workOrderListM
接收人下“类型”	名称和表达式
参与者下“类型”	表达式
参与者下“取值”	“变量”下的“assignedFme” **说明** 请直接从全局上下文拖拽“assignedFme”到“取值”下，请勿手动输入，手动输入的值系统可能不识别。

4. 配置“接单”图元。

选中“接单”图元，单击，再单击“全局上下文”，并展开全局上下文，配置“维修人员接单”图元，详细配置如表2-37所示。

说明

请直接从全局上下文拖拽“statusInfo”、“id”、“status”到输入输出参数对应“源”或“目标”下，请勿手动输入，手动输入的值系统可能不识别。

图 2-261 配置“接单”图元

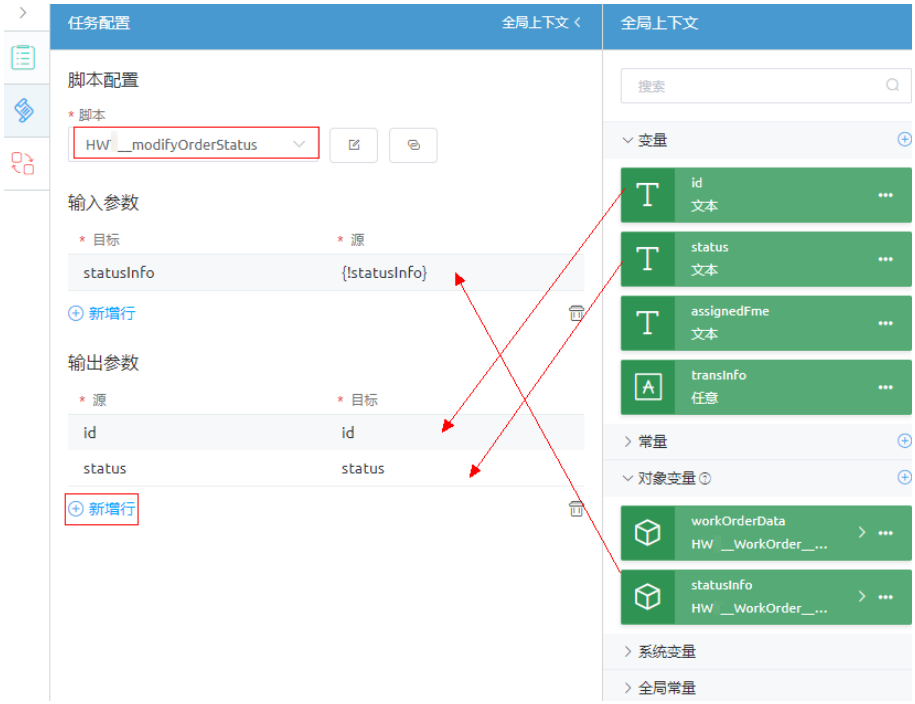


表 2-38 接单图元配置

参数项	值
脚本	***HW***__modifyOrderStatus
输入参数“statusInfo”	对象变量下“statusInfo”
输出参数“id”	变量下“id”
输出参数“status”	变量下“status”

5. 配置“维修人员关单”图元。

   选中“维修人员关单”图元，单击  ，设置“任务标题”为“维修人员关单”，“任务描述”为“维修人员任务列表”，“渲染类型”为“标准页面”，“页面”设置为“***HW***__workOrderListM”，并设置“当前泳道中的上一个任务被分配的人员”，如图2-262所示。

图 2-262 配置“维修人员关单”图元

6. 配置“关闭工单”图元。

选中“关闭工单”图元，单击，再单击“全局上下文”，并展开全局上下文，配置“关闭工单”图元，详细配置如**表2-37**所示。

说明

请直接从全局上下文拖拽“statusInfo”、“id”、“status”到输入输出参数对应“源”或“目标”下，请勿手动输入，手动输入的值系统可能不识别。

图 2-263 拖拽关闭工单输入输出值

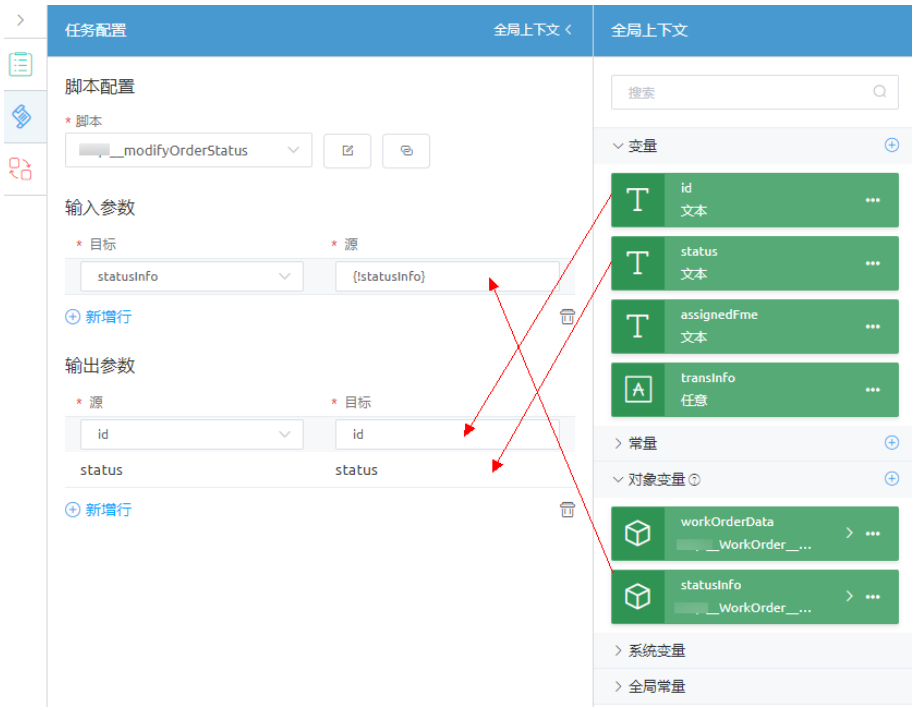
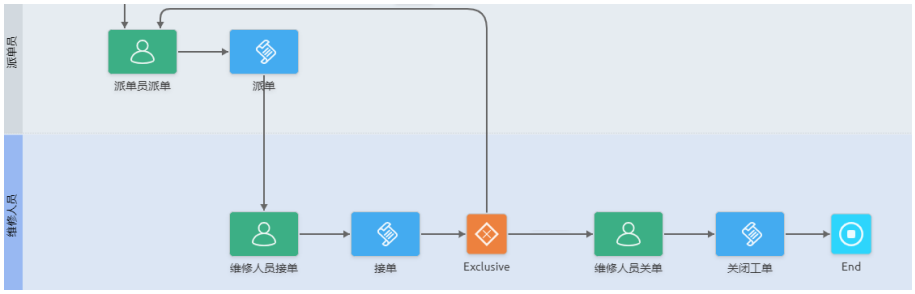


表 2-39 接单图元配置

参数项	值
脚本	***HW***__modifyOrderStatus
输入参数“statusInfo”	对象变量下“statusInfo”
输出参数“id”	变量下“id”
输出参数“status”	变量下“status”

步骤10 拖拽图元直接的关系连线，具体如图2-264所示。

图 2-264 添加图元之间连线



步骤11 配置“排他网关”图元连线属性。

1. 选中“派单员派单”与“排他网关”之间的连线，单击鼠标右键，选择“配置”，设置“标签”为“拒单”，再单击“新增行”，拖拽变量“status”到“资源”下，设置“比较符”为“==”，“值”为“**"拒单"**”。

图 2-265 选中排他网关图元连线

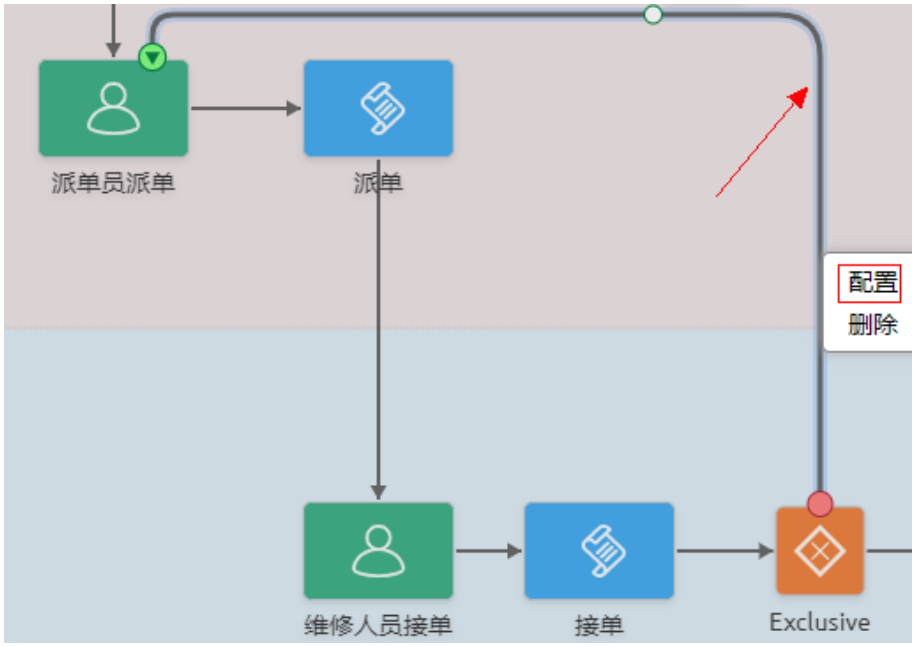


图 2-266 配置“拒单”连线

2. 选择“排他网关”图元与“维修人员关单”图元之间的连线，参考上一步，设置连线，设置“标签”为“接单”，再单击“新增行”，**拖拽**变量“status”到“资源”下，设置“比较符”为“==”，“值”为“**"接单"**”。

图 2-267 配置“接单”连线

步骤12 单击页面上方的，再单击，启用BPM。

----结束

2.7.8 验证工单管理功能

2.7.8.1 挂载前端页面

应用开发完成后，要将页面挂载到导航条上，作为应用菜单，在应用预览时，查看应用相关页面。应用菜单仅支持开发者号的管理员查看，业务号只能查看对应权限的页面。

操作步骤

步骤1 在经典版应用开发页面，单击左侧导航栏下方的“配置”。

图 2-268 应用配置入口

步骤2 在“导航条”页签，单击“菜单树”右侧的“+”，选择“添加页签”。

图 2-269 导航条添加页签

步骤3 定义“设备管理”页签。

在“添加页签”弹窗中，设置以下信息，然后单击“保存”。

- 页面类型：设置为“标准页面页签”。
- 标签：设置为“设备管理”。
- 名称：设置为“equipmentManage”。
- 页面：设置为“***HW***__equipmentManage”。

图 2-270 设备管理页签

步骤4 请按照以上方式，创建表2-40的导航菜单。

表 2-40 导航菜单

页签类型	标签	名称	页面
标准页面页签	工单管理	manageWorkOrderList	***HW***__manageWorkOrder
标准页面页签	工单列表（客服人员）	workOrderList	***HW***__workOrderList
标准页面页签	工单列表（派单员）	dispatchWorkOrder	***HW***__dispatchWorkOrder

页签类型	标签	名称	页面
标准页面页签	工单列表（维修人员）	workOrderListM	***HW***__workOrderListM

步骤5 （可选）设置导航的布局及颜色。

您可以根据自己的喜好，设置导航的布局及颜色，默认导航菜单是靠左，蓝色。

如图2-271所示，单击红框中的布局，即可将导航菜单设置为横向菜单，然后单击“保存”。

图 2-271 切换为横向导航

----结束

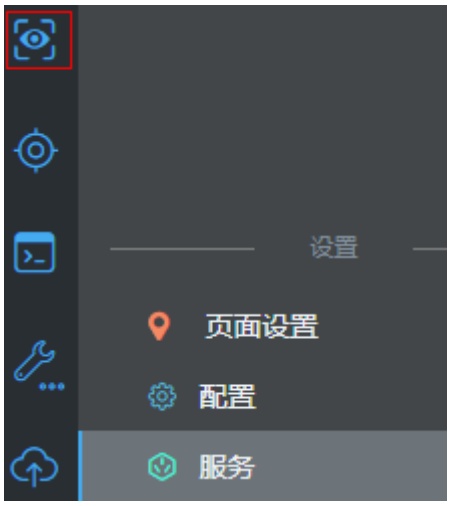
2.7.8.2 验证

本节主要是通过在不同的页面上操作工单流程，验证工单管理功能中的各个功能。

操作步骤

步骤1 在经典版应用开发页面，单击左侧导航栏下方的  ，进入应用预览页面。

图 2-272 查看应用



步骤2 验证创建工单。

1. 选择“工单列表（客服人员）”，然后在工单列表（客服人员）页面，单击“创建工单”，进入创建工单页面。
2. 在“设备名称”下拉框中选择一个设备，检查“设备详情”区域显示了对应的设备详情。显示正确，则说明组件与模型的绑定，以及下拉框“数据改变”事件执行正确。

   说明

   如果“设备名称”下，没有设备名，请先在“设备管理”页面，单击“新增行”进行添加。

3. 检查是否正常提交工单。

   填写工单信息， 单击“提交”。如果页面跳转到工单列表页面，且在查询结果中，显示新创建的工单，则说明提交按钮的“点击”事件执行正确。

   如果页面跳转到工单列表页面，且在查询结果中显示新创建的工单，则说明验证成功。

图 2-273 验证结果样例

步骤3 验证派单功能。

1. 单击“工单列表（派单员）”，进入“工单列表（派单员）”页面，查看页面是否显示上一步创建的工单，且每条记录后都有派单图标。
2. 选择上一步新建的一条工单记录，并单击该记录的 派发，弹出处理工单弹窗。

图 2-274 状态为“待派单”工单记录

3. 在处理工单弹窗中，设置下拉框“选择工程师”为当前租户账号，然后单击“确定”，返回“工单列表（派单员）”页面。

   查看工单记录的“状态”、“当前处理人”是否已更新为“待接单”、当前租户名。如果已更新，说明派单流程正常。如未改变请检查“派单”按钮上事件代码以及“处理工单”弹窗中数据绑定及事件代码。

步骤4 验证待接工单。

1. 进入“工单列表（维修人员）”页面，检查系统显示派单员刚刚派的工单，如图2-275所示。

   如未正常显示请检查当前页面标签组件的属性值绑定及页面事件代码。

图 2-275 待处理工单

2. 单击“处理”按钮，进入处理工单弹窗，“选择下一步操作”设置为“接单”，单击“提交”按钮。

   处理完成后，返回“待处理工单”页面，查看“状态”是否已经更新为“处理中”。

   如果有多条工单，“待处理”状态的工单，优先显示，“处理中”的工单可能会显示在页面的下面，工单状态改变后，可以拖动滚动条查找该工单。

   说明

   如果在“选择下一步操作”中，选择了“拒单”，流程将返回“派单员”处，状态将变为“待派单”。

图 2-276 处理工单-关单

图 2-277 查看处理中的工单

3. 单击“处理中”工单的“处理”按钮，选择“关单”，单击“提交”，返回页面后，查看该条工单的状态是否变成“关闭”，如果已关闭，则说明关单流程正常。

图 2-278 关闭工单

步骤5 参考以上步骤，验证工单管理页面。

1. 在应用菜单中，选择“工单管理”页面，进入工单管理，查看页面工单列表中的工单信息。
2. 进行“创建工单”、派单及删除工单操作，验证工单管理功能。

----结束

2.8 用户管理功能开发

2.8.1 背景与知识

“用户管理”功能包括业务新增业务用户、查看删除业务用户、添加业务用户权限集三部分。

在业务场景中，会区分不同业务用户，业务用户对应了不同的用户权限，本示例应用中包含的业务用户，即是使用设备管理应用的用户，分别是客服人员、派单员及维修人员。

在AstroZero开发的应用的所有业务用户，最终都会存储到平台的业务用户表中，即PortalUser表。业务用户在登录应用后的访问权限是通过Profile进行控制管理的，可基于平台预置的Portal User Profile进行自定义业务用户权限。

图 2-279 业务用户

2.8.2 开发业务逻辑

2.8.2.1 创建用户注册脚本

通过创建用户注册脚本，完成添加业务用户账户的业务逻辑，以供“业务用户注册”页面调用。

前提条件

参考**如何登录经典版环境配置**中操作，登录经典版环境配置，将“系统管理 > 系统参数 > 内置系统参数”中的参数“bingo.security.sensitive.data”取值修改为“是”（如果为否，通过脚本创建出来的业务用户无法正常登录）。

图 2-280 修改“bingo.security.sensitive.data”的值为“是”

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“User”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“registerPortalUser”，单击“添加”。

步骤4 在代码编辑器中，插入如下脚本代码。

```
import * as buffer from "buffer";
import * as crypto from "crypto";
```

```
import * as db from "db";
import * as context from 'context';
import * as http from 'http';
import * as permission from 'permission';

//定义入参结构，包括注册账号的用户名、密码和角色，为必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'string' })
    username: string;
    @action.param({ type: 'String', required: true, label: 'string' })
    password: string;
    @action.param({ type: 'String', required: true, label: 'string' })
    role: string;
}
//定义出参结构，出参包含1个参数，portaluser的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    msg: string;
}
//使用数据对象PortalUser
@useObject(['PortalUser'])
@action.object({ type: "method" })
export class RegisterPortalUser {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public registerPortalUser(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作pu为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let s = db.object('PortalUser');

            let saltedPassword = salt(input.password);
            let userMsg = {
                "usrName": input.username,
                "name": input.username,
                "userPassword": saltedPassword['saltedPassword'],
                "passwordSalt": saltedPassword['salt'],
                "userType": input.role
            };

            let userId = s.insert(userMsg);
            if (userId) {
                out.msg = "注册成功！";
            } else {
                error.name = "USERERROR";
                error.message = "注册失败！";
                throw error;
            }

        } catch (error) {
            if (error.name == "405230618") {
                error.message = "该用户名已注册！"
            }
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}

function _salt(password: string, saltBuf: buffer.Buffer, encoding: buffer.Encoding = buffer.Encoding.Base64):
string {
    const passwordBuf = buffer.from(password)
    const crypt = crypto.pbkdf2(passwordBuf, saltBuf, 1000, 32, crypto.Hashs.SHA1)
    return crypt.toString(encoding)
}

function salt(password: string, encoding: buffer.Encoding = buffer.Encoding.Base64): object {
```

```
    const saltBuf = crypto.randomBytes(6)
    const saltedPassword = _salt(password, saltBuf, encoding)
    return {
        salt: saltBuf.toString(encoding),
        saltedPassword: saltedPassword
    }
}
```

步骤5 单击编辑器上方的  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的  ，执行脚本。
2. 在界面底部输入测试数据，单击测试窗口右上角  执行图标。

```
{
    "username": "test_cs",
    "password": "***",
    "role": "cs"
}
```

执行成功，会在“输出”页签返回查询结果。

```
{
    "msg": "注册成功！"
}
```

步骤7 测试成功，单击编辑器上方的  ，启用发布脚本。

----结束

2.8.2.2 创建用户登录脚本

用户登录脚本是**组装“业务用户登录”页面**中，自定义登录组件调用的业务逻辑，之前创建的登录页只有前端页面，本节将创建完整的登录业务逻辑。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“User”目录中，将鼠标放在“Script”上，单击界面上出现的“+”，在弹出菜单中选择“脚本”。

步骤3 在弹窗中，选中“创建一个新脚本”，在“名称”文本框中输入“login”，单击“添加”。

步骤4 在代码编辑器中，插入如下脚本代码。

```
import * as buffer from "buffer";
import * as crypto from "crypto";
import * as db from "db";
import * as context from 'context';
//定义入参结构，包括账号的用户名、密码为必填字段,验证码为非必填字段
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'string' })
    username: string;
    @action.param({ type: 'String', required: true, label: 'string' })
    password: string;
    @action.param({ type: 'String', required: true, label: 'string' })
    captcha: string;
}
//定义出参结构，出参包含5个参数，登录结果和用户角色
@action.object({ type: "param" })
export class ActionOutput {
```

```
    @action.param({ type: 'String' })
    msg: string;
    @action.param({ type: 'String' })
    username: string;
    @action.param({ type: 'String' })
    userId: string;
    @action.param({ type: 'String' })
    captcha: string;
    @action.param({ type: 'String' })
    profile: string;
}
//使用数据对象PortalUser
@useObject(['PortalUser'])
@action.object({ type: "method" })
export class Login {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public login(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            out.captcha = input.captcha;
            let s = db.object('PortalUser');
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "usrName",
                    "operator": "eq",
                    "value": input.username
                }]
            };
            let user = s.queryByCondition(condition);
            if (user && user.length == 1) {
                if (validate(user[0].passwordSalt, user[0].userPassword, input.password)) {
                    out.msg = "登录成功！";
                    out.username = user[0].usrName;
                    out.userId = user[0].id;
                    out.profile = user[0].userType;
                } else {
                    out.msg = "密码错误！";
                }
            } else {
                out.msg = "用户不存在！";
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
            out.msg = error.message;
        }
        return out;
    }
}

function _salt(password: string, saltBuf: buffer.Buffer, encoding: buffer.Encoding = buffer.Encoding.Base64):
string {
    const passwordBuf = buffer.from(password)
    const crypt = crypto.pbkdf2(passwordBuf, saltBuf, 1000, 32, crypto.Hashs.SHA1)
    return crypt.toString(encoding)
}

function validate(salt: string, userSaltedPassword: string, password: string, encoding: buffer.Encoding =
buffer.Encoding.Base64): boolean {
    const saltBuf = buffer.from(salt, encoding);
    const saltedPassword = _salt(password, saltBuf, encoding);
    return saltedPassword === userSaltedPassword
}
```

步骤5 单击编辑器上方的  ，保存脚本。

步骤6 测试脚本能否正常执行。

1. 单击编辑器上方的，执行脚本。
2. 在界面底部输入测试数据，单击测试窗口右上角执行图标，其中“*test_cs*”、变量“*{XXXXXXXX}*”为**用户注册**脚本中测试数据。

```
{
 "username": "test_cs",
 "password": "{XXXXXXXX}",
 "captcha": ""
 }
```

执行成功，会在“输出”页签返回查询结果。

```
{
   "captcha": "",
   "msg": "登录成功！",
   "profile": "cs",
   "userId": "10gg0XXXXXXXXXXXX",
   "username": "test_cs"
}
```

步骤7 测试成功，单击编辑器上方的，启动发布脚本。

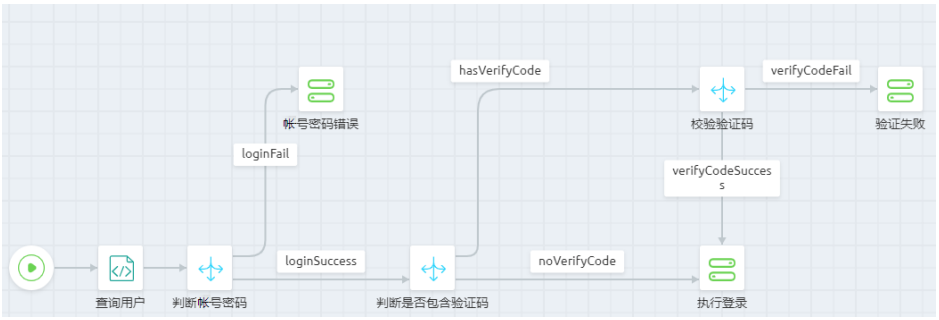
----结束

2.8.2.3 创建用户登录服务编排

用户登录服务编排通过调用脚本、编排图元等操作实现用户登录完整逻辑。

实现原理

图 2-281 用户登录服务编排大致设想



如**图2-281**所示，用户登录服务编排业务逻辑实现过程如下：

1. 通过调用“用户登录”脚本，查询登录账户密码，然后使用“决策”图元进行判断，判断当前登录的账号密码是否正确。
2. 如果判断账户密码错误，直接执行“账户密码错误”，判断账户密码正确，则使用“决策”图元，继续判断是否有验证码。
3. 如果判断当前登录没有验证码，则直接执行登录，判断当前有验证码，则继续判断验证码是否正确。
4. 如果判断验证码正确，则执行登录操作，判断验证码错误，则执行验证失败。

操作步骤

“用户登录”服务编排开发的大致过程：先拖拽1个脚本图元，3个决策图元以及3个赋值图元，再分别配置各个图元属性，然后配置各个图元之间连线类型，最后保存启用。

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“User”目录中，将鼠标放在“Flow”上，单击界面上出现的“+”，在弹出菜单中选择“服务编排”。

步骤3 选中“创建一个新的服务编排”，在“标签”和“名称”文本框中输入“login”，并设置类型为“Autolaunched Flow”，单击“添加”。

步骤4 定义服务编排用到的变量。

1. 单击，展开全局上下文，再单击“变量”后的，设置参数名称为“username”，如图2-282所示。

图 2-282 新增变量

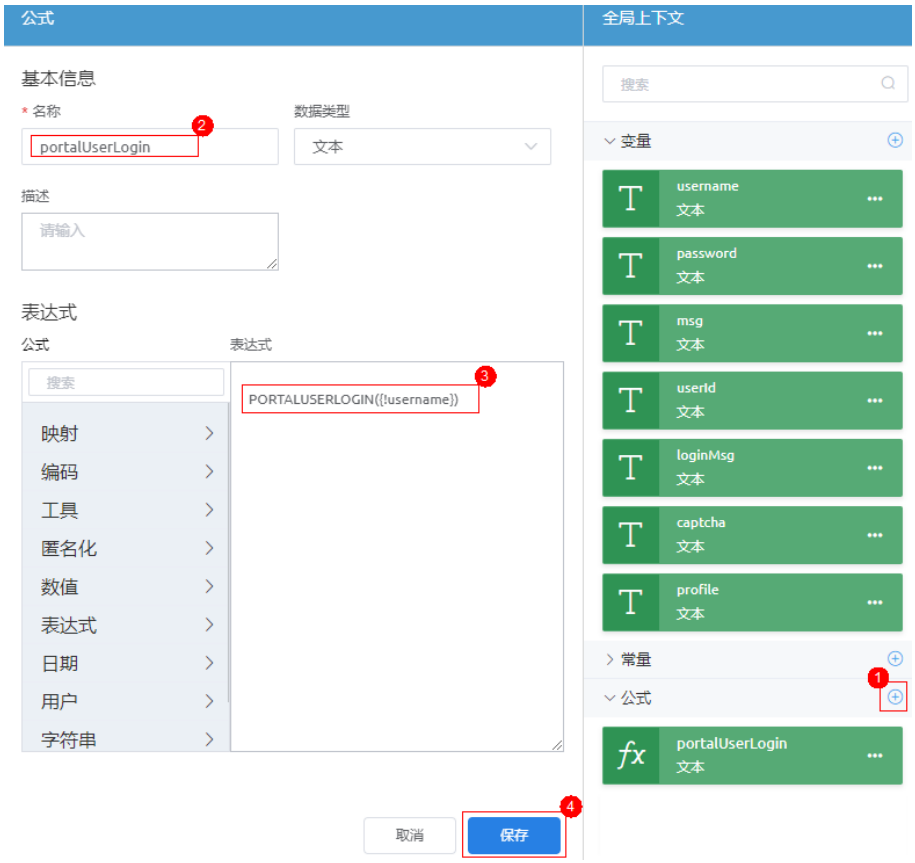
2. 重复上一步，定义其他变量。
需要定义的变量如表2-41所示。

表 2-41 服务编排变量说明

变量名称（唯一标识）	数据类型
username（上一步已创建）	文本
password	文本
msg	文本
userId	文本
loginMsg	文本
captcha	文本
profile	文本

3. 单击“公式”后的，在左侧公式弹窗中，设置“名称”为“portalUserLogin”，“表达式”为“PORTALUSERLOGIN({!username})”，单击“保存”。

图 2-283 添加公式变量“portalUserLogin”



4. 参考上一步，创建公式变量“verifyCode”，需要定义的变量如表2-42所示。

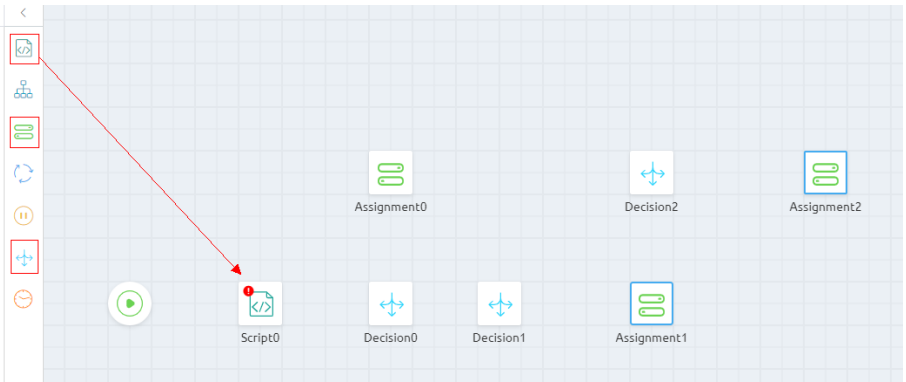
表 2-42 公式变量说明

名称	表达式
verifyCode	VERIFYCODEWITHTYPE({!captcha},"login")

步骤5 拖拽图元到服务编排画布，并配置图元的基本属性。

1. 从图元区分别拖拽脚本（1个）、决策（3个）、赋值（3个）图元到画布中，排列，如图2-284所示。

图 2-284 拖拽脚本（1 个）、决策（3 个）、赋值（3 个）图元



2. 选中“Script0”图元，在右侧基本信息中，设置“标签”为“查询用户”。
3. 参考**步骤5.2**，设置其他图元的“标签”属性，具体值如**表2-43**所示。

表 2-43 图元基本属性

名称（变量唯一标识，不需要修改）	标签
Script0（上一步已设置）	查询用户
Decision0	判断账号密码
Decision1	判断是否包含验证码
Decision2	校验验证码
Assignment0	账号密码错误
Assignment1	执行登录
Assignment2	验证失败

图 2-285 修改后图元

步骤6 配置“查询用户”脚本图元。

1. 单击  ，指定图元对应的脚本名称（***HW***__login），并配置脚本的输入输出参数。
2. 单击“全局上下文”，显示变量列表，从“变量”中，拖拽“username”、“password”、“captcha”到“输入参数”下对应的“源”输入框中。
3. 在“输出参数”下，单击5次“新增行”，依次添加下拉选项中的输出参数字段，并从“变量”中**拖拽**相应的字段到“目标”输入框下，字段与变量对应关系如**图2-286**所示。

说明

请直接从全局上下文拖拽“变量”到对应的输入输出参数下，请勿手动输入，手动输入的值系统可能不识别。

图 2-286 拖拽脚本的输入输出参数

步骤7 配置“判断账号密码”决策图元。

1. 选择“判断账号密码”图元，在右侧单击图标，修改“默认”的“名称”为“loginFail”。

图 2-287 修改“默认”结果名称

2. 单击“新增”，增加一个可编辑的结果，修改结果为“loginSuccess”，在“可视”下单击“新增行”，并**拖拽**变量中的“msg”到“资源”下，设置“比较符”为“==”，“值”为“**"登录成功！"**”。

说明

- 请直接从全局上下文拖拽变量“msg”到“资源”下，请勿手动输入，手动输入的值系统可能不识别。
- **"登录成功！"**需要与“login”脚本中的输出参数一致。

图 2-288 修改可编辑的结果

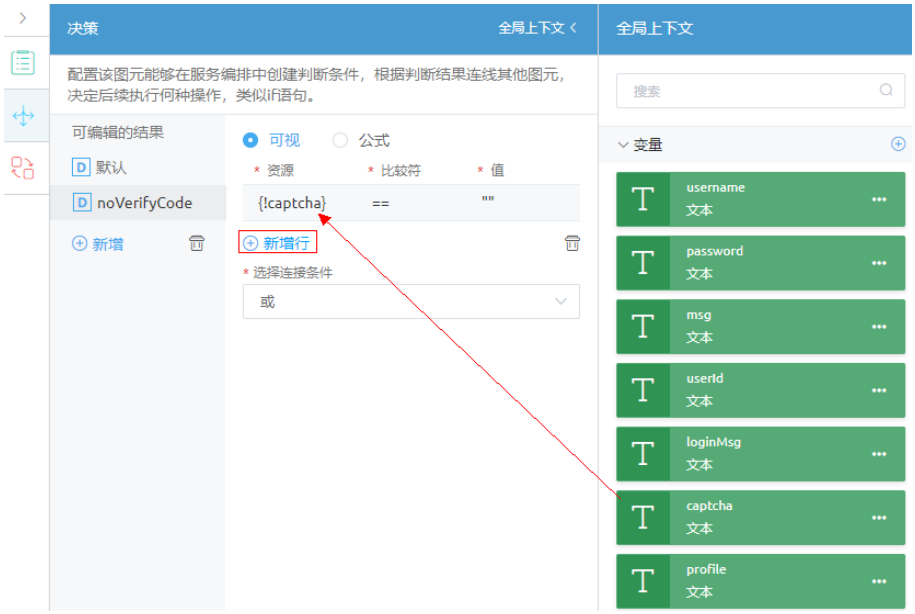
步骤8 配置“判断是否包含验证码”决策图元。

1. 选择“判断是否包含验证码”图元，在右侧单击图标，修改“默认”的“名称”为“hasVerifyCode”。

图 2-289 修改“默认”结果名称

2. 单击“新增”，增加一个可编辑的结果，修改结果为“noVerifyCode”，在“可视”下单击“新增行”，并**拖拽**变量中的“captcha”到“资源”下，设置“比较符”为“==”，“值”为“""”。

图 2-290 修改可编辑的结果



步骤9 配置“校验验证码”决策图元。

1. 选择“校验验证码”图元，在右侧单击图标，修改“默认”的“名称”为“verifyCodeFail”。

图 2-291 修改“默认”名称

2. 单击“新增”，增加一个可编辑的结果，修改结果为“verifyCodeSuccess”，在右侧选择“公式”，并从全局上下文中，**拖拽**“verifyCode”到“公式”下。

图 2-292 修改可编辑的结果

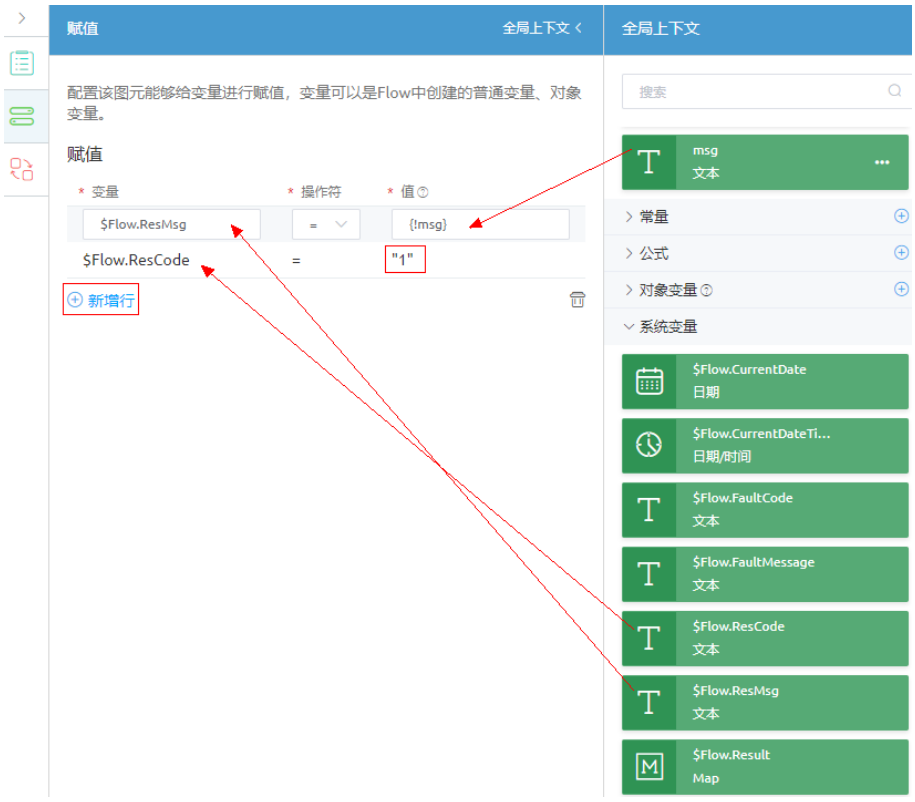
步骤10 配置“账号密码错误”赋值图元。

选择“账号密码错误”图元，在右侧单击图标，单击“新增行”，从全局上下文的“系统变量”中，拖拽“$Flow.ResMsg”到“赋值”下，并设置“操作符”为“=”，拖拽“msg”到“值”；然后再拖拽“系统变量”下的“$Flow.ResCode”到“赋值”的“变量”下，设置“操作符”为“=”，设置“值”为“**"1"**”。

说明

请直接从全局上下文拖拽变量到“赋值”下的对应位置，请勿手动输入，手动输入的值系统可能不识账。

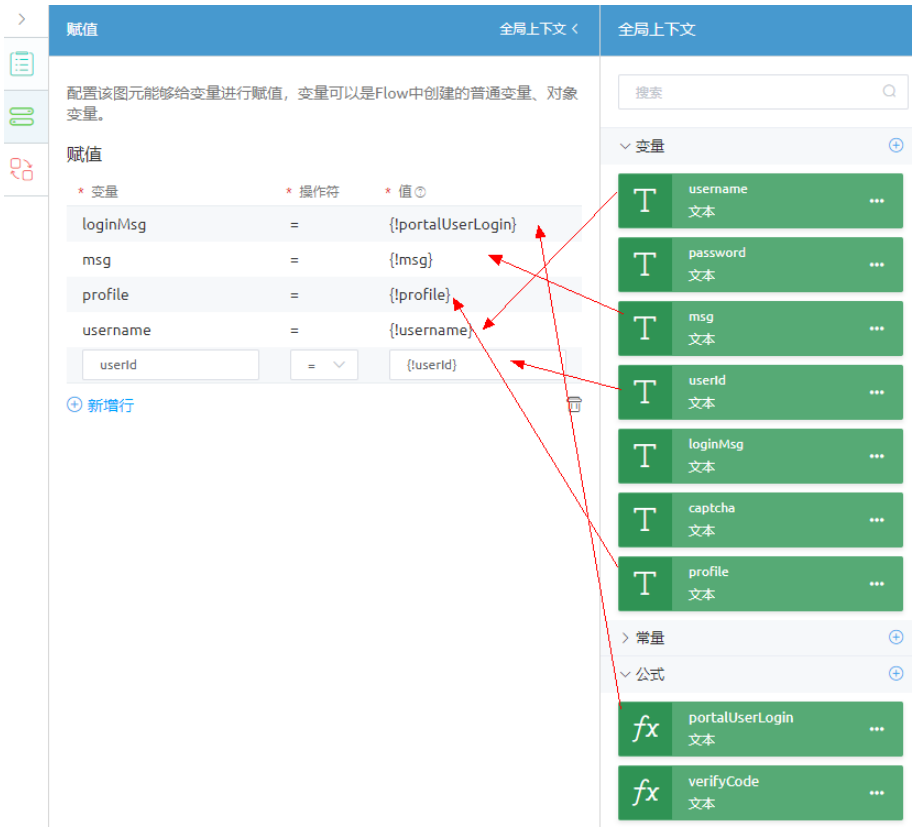
图 2-293 配置“账号密码错误”图元



步骤11 配置“执行登录”赋值图元。

1. 选择“执行登录”图元，在右侧单击图标，单击5次“新增行”。
2. 从全局上下文，拖拽“msg”等字段到“赋值”的“变量”下，并设置“操作符”为“=”，然后再拖拽“值”下的各个值，具体字段对应关系，如图2-294所示。

图 2-294 拖拽“执行登录”赋值的变量及值



说明

请直接从全局上下文拖拽变量到“值”下的对应位置，请勿手动输入，手动输入的值系统可能不识别。

表 2-44 变量与值对应关系

变量	操作符	值
loginMsg	=	portalUserLogin
msg	=	msg
profile	=	profile
username	=	username
userId	=	userId

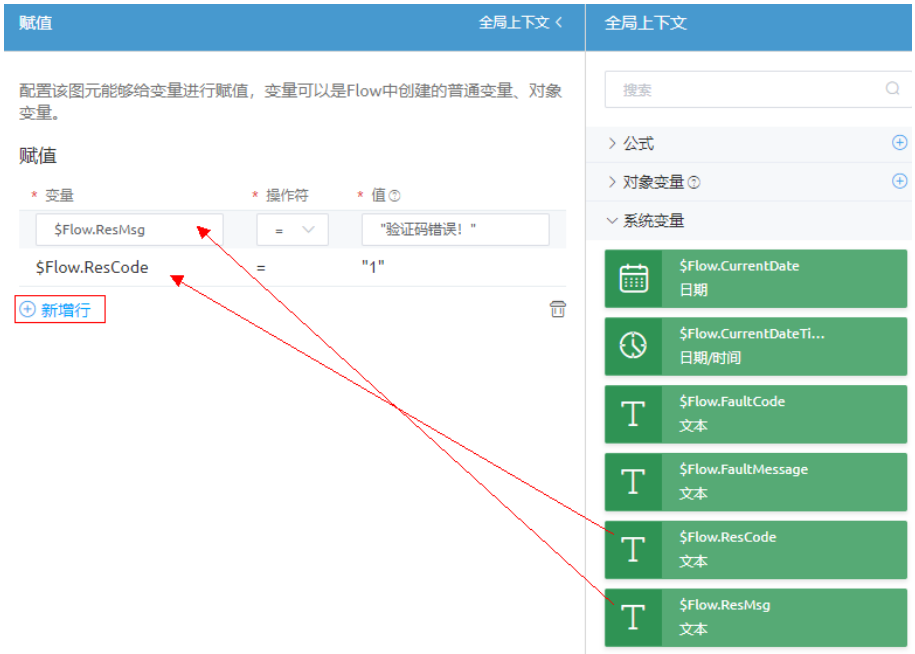
步骤12 配置“验证失败”赋值图元。

选择“验证失败”图元，在右侧单击图标，单击“新增行”，从全局上下文“系统变量”，**拖拽**“$Flow.ResMsg”、“$Flow.ResCode”到“赋值”下，并设置操作符为“=”，分别设置“值”为“**"验证码错误！"**”、“**"1"**”。

表 2-45 赋值

变量	操作符	值
$Flow.ResMsg	=	**"验证码错误！"**
$Flow.ResCode	=	**"1"**

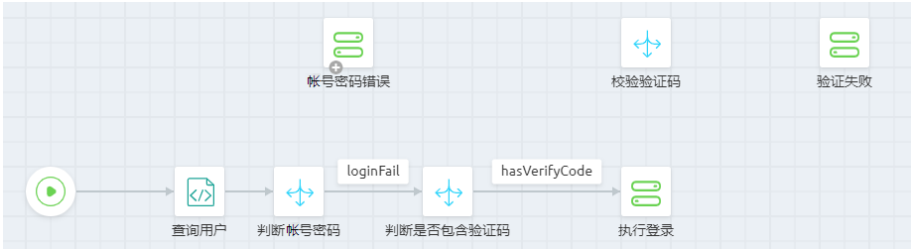
图 2-295 配置“验证失败”赋值图元



步骤13 拖拽图元连线，并配置连线属性。

1. 在画布上，把鼠标放在起点图元图元上，从“+”拖动鼠标，在起点图元和“查询用户”图元间增加连线；即将当前脚本设置为服务编排的起始节点。
2. 依次在“查询用户”、“判断账号密码”、“判断是否包含验证码”、“执行登录”图元直接拖拽连线。

图 2-296 拖拽连线



3. 单击“判断账号密码”与“判断是否包含验证码”图元之间的连线，在右侧属性单击，在“连线”中修改“连线类型”为“loginSuccess”。

图 2-297 选中连线

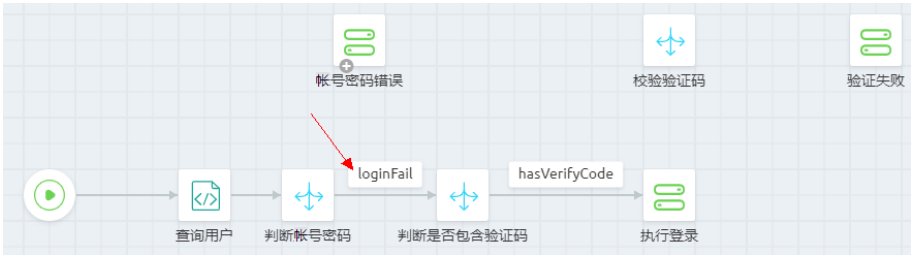
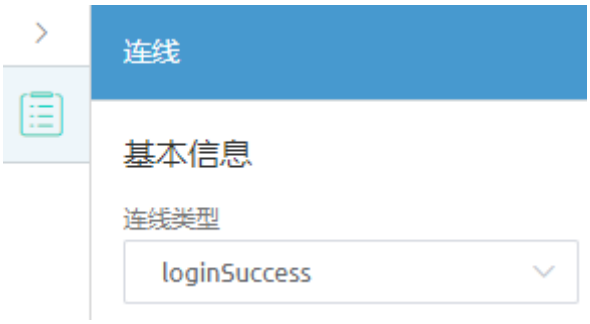


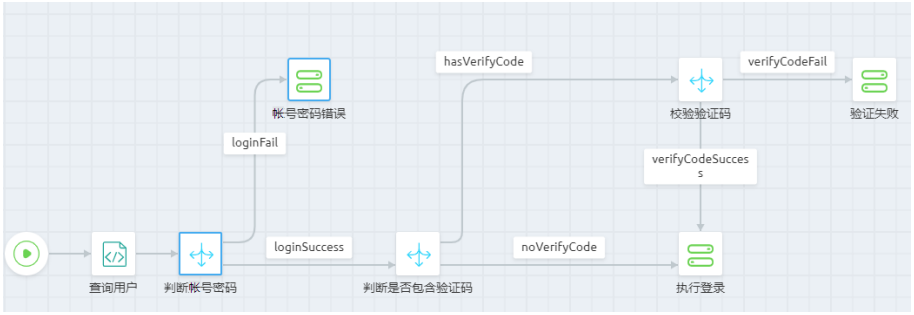
图 2-298 修改连线类型



4. 单击“判断是否包含验证码”与“执行登录”图元之间的连线，在右侧属性单击，在“连线”中修改“连线类型”为“noVerifyCode”
5. 从“判断账号密码”图元上拖拽一条连线到“账号密码错误”图元。
6. 从“判断是否包含验证码”图元上拖拽一条连线到“校验验证码”图元。
7. 从“校验验证码”图元上拖拽一条连线到“验证失败”图元。
8. 从“校验验证码”图元上拖拽一条连线到“执行登录”图元，并设置该连线的“连线类型”为“verifyCodeSuccess”。

   连线拖拽完成，如图2-299所示。

图 2-299 拖拽图元连线



步骤14 定义服务编排的输入、输出参数，并保存服务编排。

1. 鼠标在画布空白处点一下，单击右侧，设置服务编排的输入输出参数，如图2-300所示。

图 2-300 拖拽服务编排的输入输出参数

2. 单击服务编排页面上方的，保存服务编排。
   系统会弹出窗口，显示编译结果。

步骤15 测试服务编排能否正常执行。

1. 单击服务编排编辑器上方的，执行服务编排。
2. 在“Flow Run”界面中输入测试数据，单击“运行”。其中，“*test_cs*”、“*{XXXXXXXX}*”为**用户注册**脚本中测试数据。

```
{
 "username": "test_cs",
 "password": "{XXXXXXXX}",
 "captcha": ""
 }
```

执行成功，界面上会返回设备对象中的全部信息，样例如下：

```
{
  "interviewId": "002N000000jjQ95dKbCK",
  "outputs": {
    "loginMsg": "null",
    "msg": "登录成功！",
    "profile": "cs",
    "userId": "10gg0XXXXXXXXXXXX",
    "username": "test_cs"
  }
}
```

步骤16 （可选）在服务编排编辑器单击“跟踪”，可以查看到上一步的执行日志，方便定位错误。

步骤17 测试成功，单击服务编排编辑器上方的，启用发布服务编排。

----结束

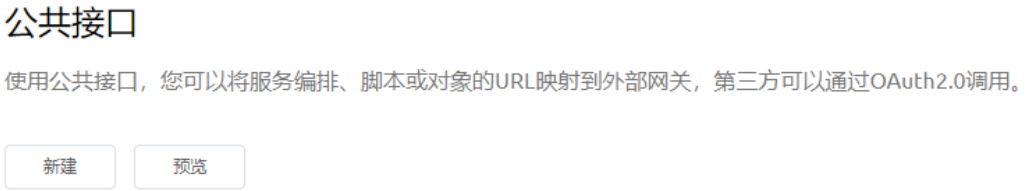
2.8.2.4 创建公共接口

步骤1 在经典版应用开发页面，单击左侧导航栏下方的“服务”，进入公共接口创建页面。

图 2-301 服务入口

步骤2 在公共接口中，单击“新建”。

图 2-302 公共接口创建



步骤3 创建“用户登录”、“用户注册”脚本对应的公共接口，详细接口信息如表2-46所示。

说明

如果在“资源”下拉框中，未找到需要关联的脚本或服务编排，请检查相关脚本和服务编排是否已启用，**加粗斜体内容**以实际命名空间前缀为准。

表 2-46 公共接口

设置操作	版本	URL	方法	类型	资源
login	1.0.0	/login	POST	服务编排	***HW***__login
registerPortalUser	1.0.0	/registerPortalUser	POST	脚本	***HW***__registerPortalUser

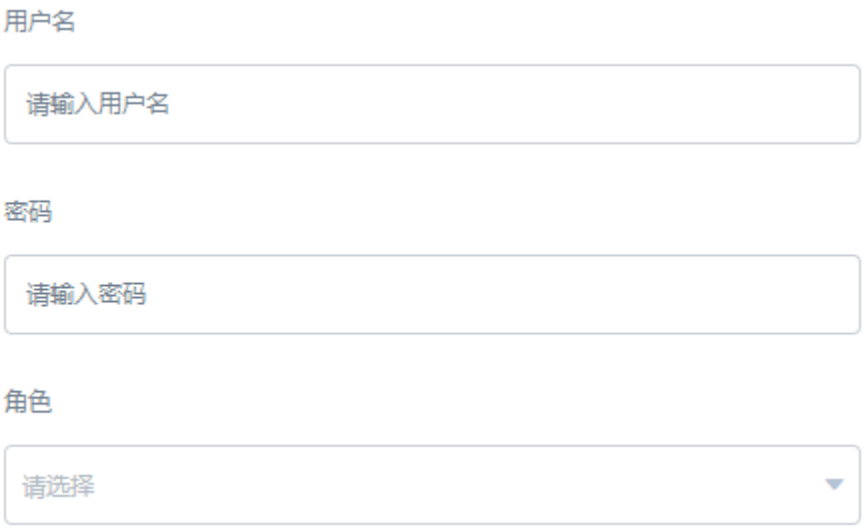
----结束

2.8.3 组装页面

2.8.3.1 组装“业务用户注册”页面

“业务用户注册”页面实际上是一个对话框，是在管理员操作“新增用户”时，弹窗的注册对话框，详细如图2-303所示。

图 2-303 业务用户注册对话框



结合页面需求，页面需要创建如下模型：

表 2-47 模型分析

模型名称	作用	详细定义
userInfo	保存注册用户信息的参数。	自定义模型，包含的计算节点如下： ● username：用户名，与输入框绑定。 ● password：密码，与输入框绑定。 ● role：角色，与下拉框绑定。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“User”目录中，鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

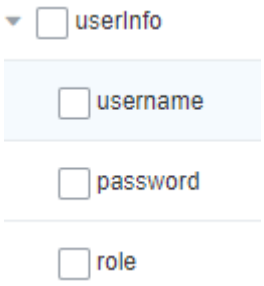
步骤3 在“标签”和“名称”文本框中输入“registerPortalUser”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击进行锁定。

步骤4 定义模型“userInfo”。

1. 在页面底部单击“模型视图”，然后在“模型视图”中，单击“新增模型”。
2. 添加自定义模型，模型名称“userInfo”，单击“下一步”。
3. 单击“新增节点”，为模型添加节点“username”、“password”、“role”字段，字段类型都采用默认的Text。单击“下一步”，再单击“确定”。

图 2-304 新增节点



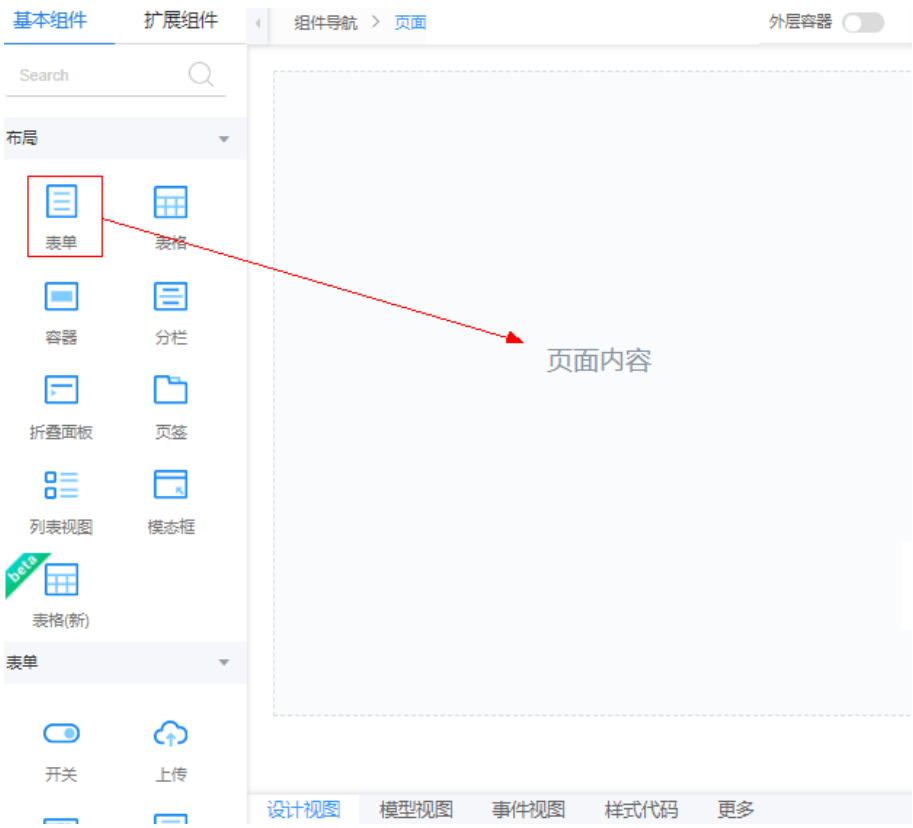
4. 单击页面上方的，保存模型。

步骤5 拖拽页面组件。

1. 单击“设计视图”，从“模型视图”切换到“设计视图”。
2. 从左侧基本组件列表区中，拖拽1个“表单”到“页面内容”中。

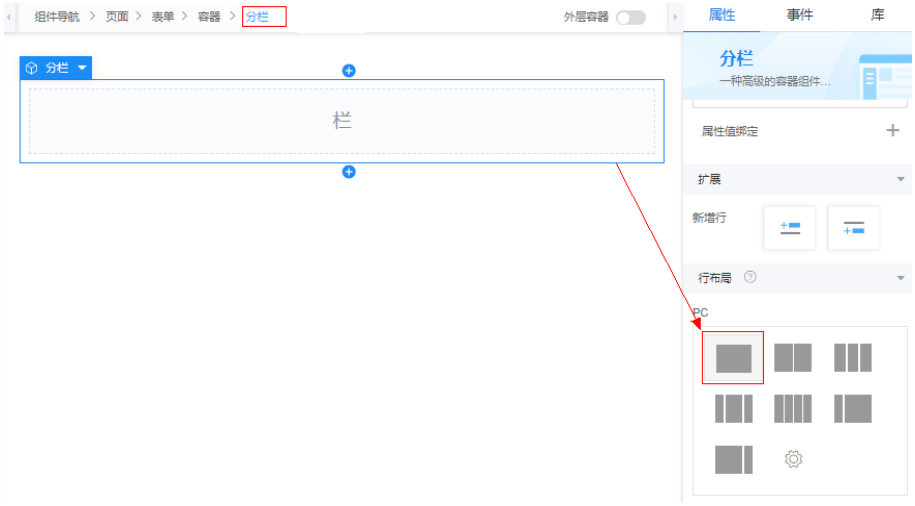
   因为当前还没有定义数据源，在“元数据表单配置向导”弹窗底部，单击“取消”，创建一个空的表单控件。

图 2-305 拖拽一个空表单



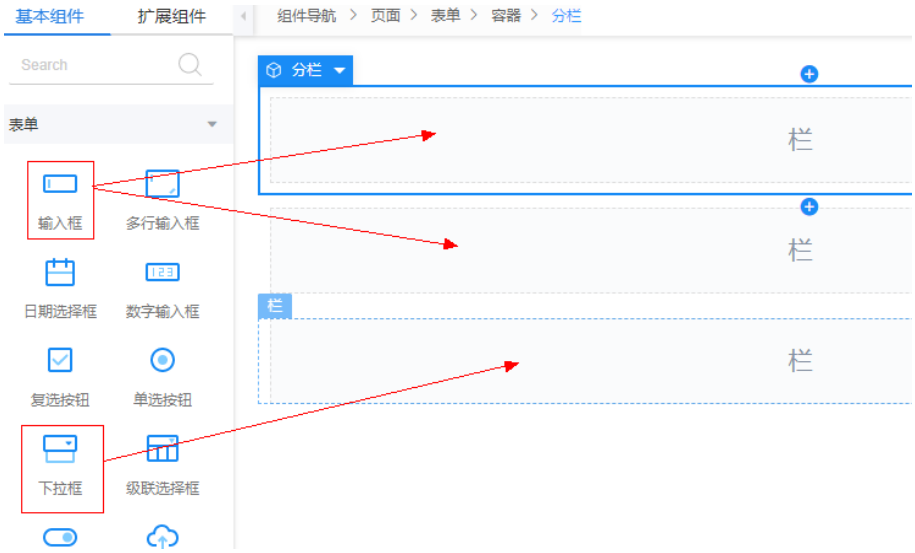
3. 拖拽1个“容器”到“表单”。
4. 拖拽1个“分栏”到“容器”。
5. 修改“分栏”为1栏（1行1列）。

图 2-306 设置分栏为 1 行 1 列



6. 选择“分栏”下的 2次，新增2个分栏。
7. 向前两个分栏，分别拖拽1个“输入框”组件，向第三个分栏中，拖拽一个“下拉框”组件。

图 2-307 拖拽输入框、下拉框



步骤6 配置组件属性，及数据绑定。

1. 设置第一个“输入框”属性。
   a. 添加值绑定。

      选中“输入框”组件，单击“数据绑定”下“值绑定”的，在弹窗中选择“ userInfo”下的“username”字段。
   b. 修改“标签”为“用户名”，“占位符”为“请输入用户名”。

2. 设置第二个“输入框”属性。
   a. 添加值绑定。

      选中“输入框”组件，单击“数据绑定”下“值绑定”的，在弹窗中选择“ userInfo”下的“password”字段。
   b. 修改“标签”为“密码”，“占位符”为“请输入密码”。
3. 设置“下拉框”属性。
   a. 添加值绑定。

      单击“数据绑定”下“值绑定”的，在弹窗中选择“userInfo”下的“role”字段。
   b. 修改“标签”为“角色”，并设置“选项”为“cs，客服人员”，“ds，派单人员”、“ms，维修人员”。

图 2-308 设置选项值

步骤7 单击页面上方的，保存设置。

步骤8 单击页面上方的，进入预览页面，查看页面是否符合预期，并单击查看是否有下拉框选项。

图 2-309 页面预览

----结束

2.8.3.2 组装“业务用户管理”页面

“业务用户管理”页面主要是通过页面对象模型与“PortalUser”标准对象绑定，并在前端页面保存新增业务用户时，调用“用户注册”逻辑，完成对该对象进行增删查的操作，实现业务用户管理功能。

结合页面需求，页面需要创建如下模型：

表 2-48 模型分析

模型名称	作用	详细定义
portalUser	与后台标准对象PortalUser绑定。	对象模型，绑定对象以下字段。 • usrName：用户名 • profile：权限 • userType：用户类型 • createdBy：创建者 • createdDate：创建日期
registerUser	与registerPortalUser公共接口绑定。	在页面获取到用户注册信息时，调用registerPortalUser公共接口，对PortalUser进行新增操作。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在“User”目录中，鼠标放在“Page”上，单击界面上出现的“+”，在弹出菜单中选择“标准页面”。

步骤3 在“标签”和“名称”文本框中输入“portalUserList”，单击“添加”。

当编辑已有标准页面时，为防止编辑时多人篡改，编辑前请单击进行锁定。

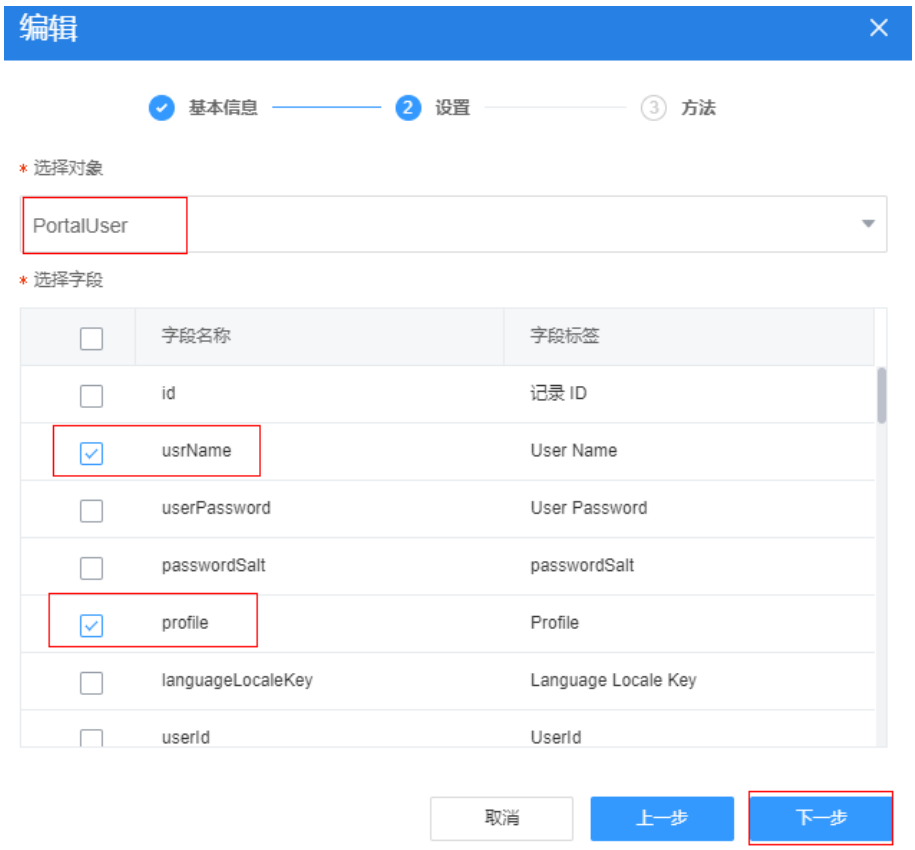
步骤4 定义对象模型“portalUser”。

1. 在页面底部单击“模型视图”，然后在“模型视图”中，单击“新增模型”。
2. 添加对象模型，模型名称“portalUser”，单击“下一步”。
3. 在“选择对象”下，选择标准对象“PortalUser”，并勾选以下字段，然后单击“下一步”，再单击“确定”。

表 2-49 选择字段

节点名称	数据类型
usrName	保持默认
profile	保持默认
userType	保持默认
createdBy	保持默认
createdDate	保持默认

图 2-310 选择对象



4. 单击页面上方的，保存模型。

步骤5 定义与API（registerPortalUser:1.0.0）关联的服务模型。

1. 在“模型视图”下，单击“新增模型”。
2. 设置模型名称为“registerUser”，来源选择“服务”，单击“下一步”。
3. “选择服务类型”为“公共接口”，“选择项目”为“设备维修管理系统”，指定模型与API“registerPortalUser”关联，单击“确定”，再单击“下一步”。

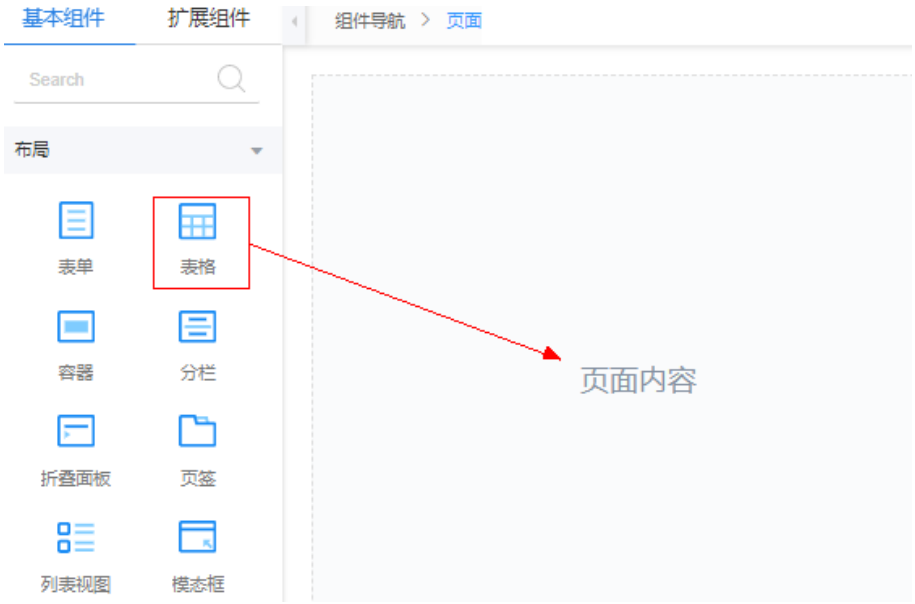
图 2-311 选择服务类型为 API

关联API后，系统会自动显示API中脚本的输入、输出参数。

4. 方法保持不变，单击“确定”。
5. 单击页面上方的，保存设置。

步骤6 切换到“设计视图”，从左侧基础组件区，拖拽一个“表格”到右侧“页面内容”中。

图 2-312 拖拽一个表格到页面



步骤7 为表格绑定模型，并设置表格查询结果区域。

1. 在“设计视图”中，选中“表格”，单击右侧“属性”页签，如图2-313所示。

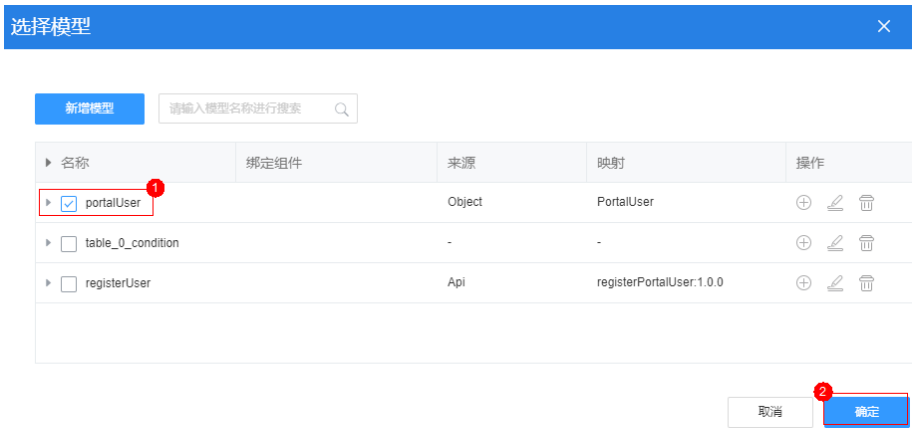
图 2-313 为表格绑定对象模型

2. 在“选择模型”对话框中，勾选“portalUser”对象模型，再单击“确定”。

须知

勾选“portalUser”后，可以将该节点下全部字段自动添加为表格的列。

图 2-314 绑定模型

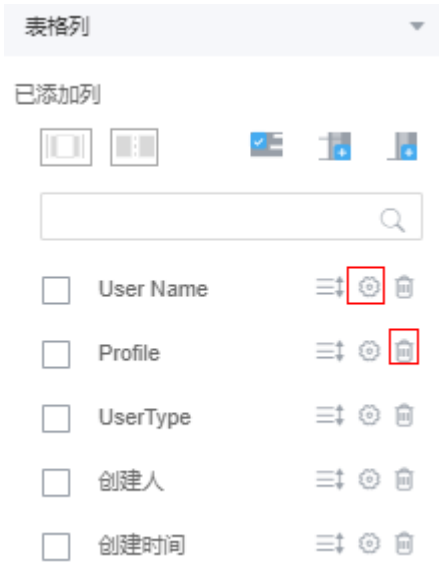


绑定对象模型后，系统自动将模型的所有字段添加为表格列，如图2-315所示。

图 2-315 绑定数据模型后的表格列

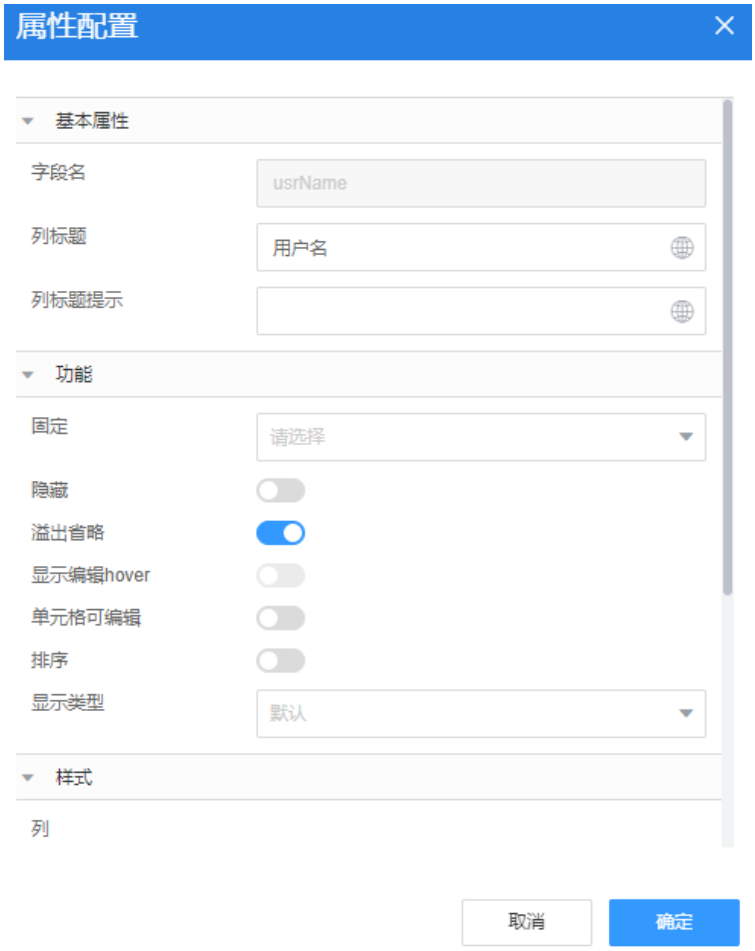
3. 选中“表格”，在右侧“已添加列”中，单击“Profile”后的，删除“Profile”字段。

图 2-316 删除字段



4. 单击“User Name”后的，修改“列标题”为“用户名”。

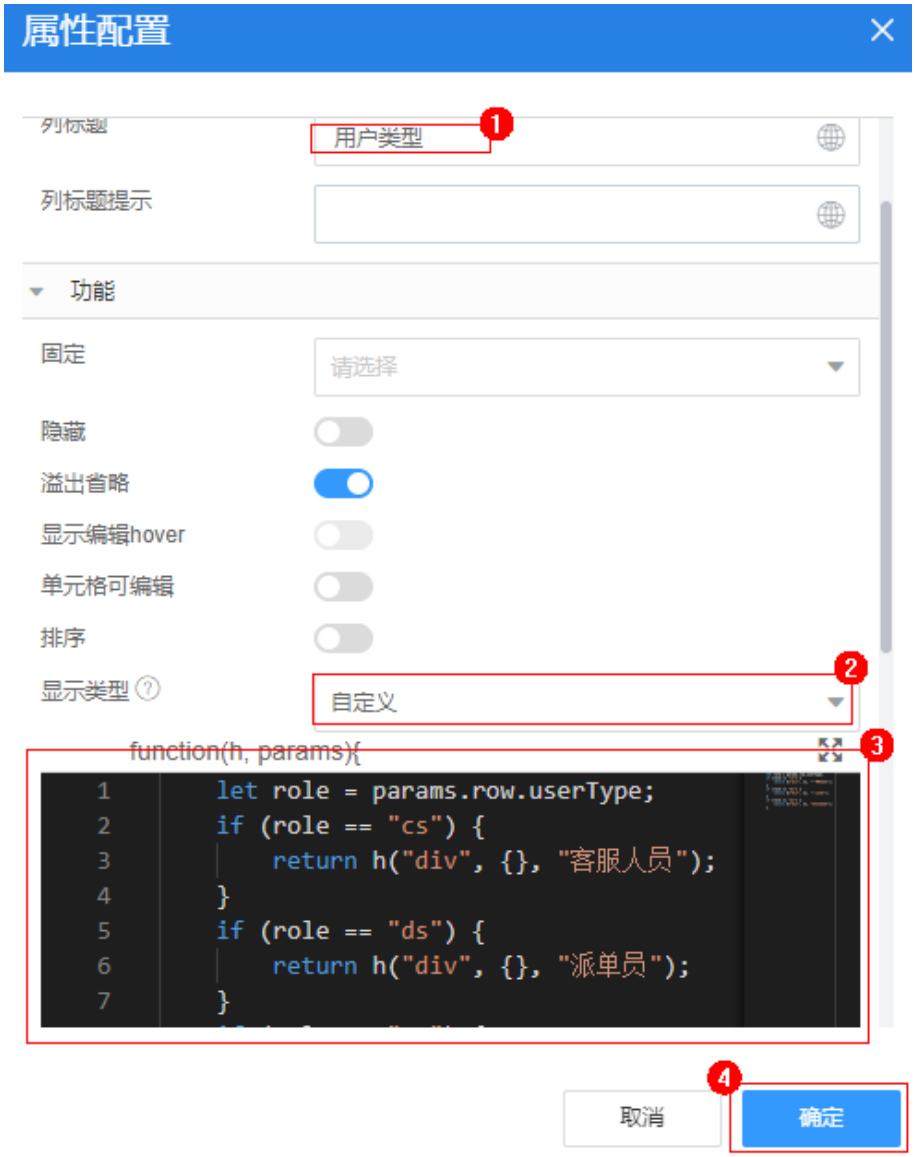
图 2-317 设置“User Name”列标题



5. 单击“UserType”后的，修改“列标题”为“用户类型”，“显示类型”设置为“自定义”，并在输入以下自定义事件代码，然后单击“确定”。

```
let role = params.row.userType;
if (role == "cs") {
    return h("div", {}, "客服人员");
}
if (role == "ds") {
    return h("div", {}, "派单员");
}
if (role == "ms") {
    return h("div", {}, "维修人员");
}
```

图 2-318 修改“UserType”列属性



6. 单击页面上方的，保存设置。

步骤8 设置工具栏区域。

1. 在左侧“设计视图”中选中“表格”，单击右侧“属性”页签“表格区块”中“工具栏”后的“添加”按钮。

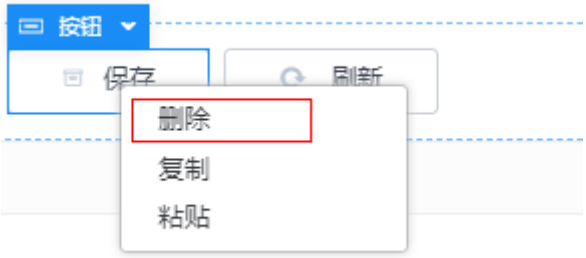
图 2-319 增加工具栏

2. 只保留“新增行”、“删除”按钮，删除其他工具栏中多余的按钮。

单击选中待删除按钮，鼠标右键单击，**请确认已选中按钮**，然后选择“删除”，如果误删其他组件，在页面上单击返回按钮即可。

图 2-320 工具栏中需要删除的按钮

图 2-321 选中待删除按钮



3. 选中“新增行”按钮，在右侧属性面板中，修改“显示名称”为“新增用户”。
4. 单击页面上方的，保存设置。

步骤9 创建查询条件区域。

1. 选中“表格”，单击右侧“属性”页签“表格区块”中“查询”后的“添加”按钮。

图 2-322 增加查询

2. 在弹窗中勾选“usrName”、“userType”，单击“确定”，表格上方自动生成查询条件区域。

图 2-323 添加查询条件

3. 拖拽一个“下拉框”组件到“用户类型”输入框下方，然后选择“用户类型”输入框，右键单击删除。
4. 配置“下拉框”组件属性。
   a. 选择“下拉框”组件，在右侧属性面板中，单击“数据绑定”，在数据绑定弹窗中，勾选“table_0_condition”的“queryInfo”下的“userType”。

      图 2-324 添加数据绑定

   b. 修改“下拉框”的“标签”为“用户类型”，并设置“选项”为“cs，客服人员”，“ds，派单人员”、“ms，维修人员”。

图 2-325 设置选项值

步骤10 添加“新增用户”按钮的“点击”事件代码。

单击“事件”页签，再单击“新增行”的，在“自定义动作”中，删除原有事件代码，输入以下事件代码，然后单击“创建”。

图 2-326 编辑事件代码


```
let _table = context.$component.table;
context.$dialog.popup(
  {
    title: '新增用户',    //弹出框标题
    page: '***HW***_registerPortalUser', // 弹出页面名称
    footerHide:false,
    width:40, //宽，小于100为百分比，大于100为px
    height:450, // 高px
    okText: '确定',
    cancelText: '取消',
    onCancel: () => {
      $model.ref("userInfo").setData({});
    }, // 取消时回调
    onOk: () => {
```

```
        // 确定时回调
        //debugger;
        let userInfo = $model.ref("userInfo").getData();
        $model.ref('registerUser').setData({inputParam: userInfo});
        $model.ref('registerUser').run().then(function (data) {
            if (data && data.msg){
                context.$message.success(data.msg);
                _table.doQuery();
            }
        }).catch(function (error) {
            console.log('error is', error);
            context.$message.error(error.resMsg);
        });
    }
  }
)
```

步骤11 单击页面上方的，保存设置。

步骤12 单击页面上方的，进入预览页面，查看页面是否符合预期。

1. 查看页面样式是否符合预期，即查看“用户类型”下拉框中，选项值是否正常。
2. 查看用户列表中是否有测试数据。

图 2-327 业务用户管理

3. 单击“新增用户”，查看是否弹出“业务用户注册”页面，如果正常显示，则说明按钮事件代码正确，如果不显示弹窗，请检查按钮事件代码。
4. 在“业务用户注册”弹窗中，输入“用户名”、“密码”，并设置“角色”，单击“确定”，是否提示“注册成功”，注册成功后，返回“业务用户管理”，查看业务用户列表中，是否显示新增用户信息。如果新增用户失败，请检查“业务用户管理”页面相关事件代码。

----结束

2.8.3.3 验证

验证用户管理功能之前，需要先将“业务用户管理”页面挂载到导航条上，应用菜单仅支持开发者账号的管理员查看，业务账号只能查看对应权限的页面。

操作步骤

步骤1 在经典版应用开发页面，单击左侧导航栏下方的“配置”。

图 2-328 应用配置入口

步骤2 在“导航条”页签单击“菜单树”右侧的“+”，选择“添加页签”。

图 2-329 添加页签

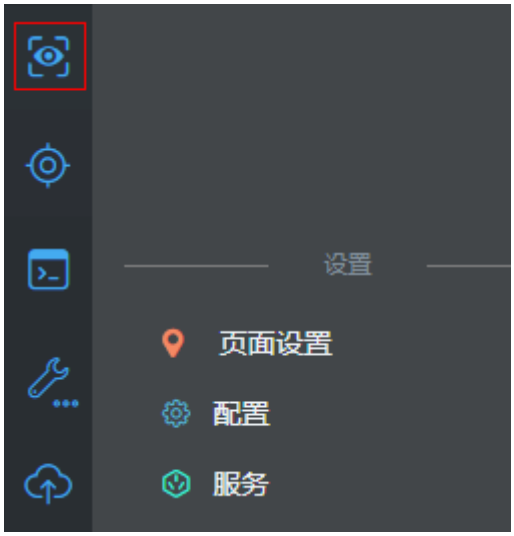
步骤3 定义“新增用户”页签。在“添加页签”弹窗中，设置以下信息，然后单击“保存”。

- 页面类型：设置为“标准页面页签”。
- 标签：设置为“新增用户”。
- 名称：设置为“addUser”。
- 页面：设置为“portalUserList”。

图 2-330 添加“新增用户”页面

步骤4 在应用左侧导航下，单击  ，进入应用预览页面。

图 2-331 查看应用



步骤5 在应用菜单上，单击“新增用户”，进入用户管理页面。

图 2-332 新增用户

步骤6 在用户管理页面，单击“新增用户”，在弹窗中设置用户名、密码及角色。

图 2-333 添加新用户

步骤7 创建完成后，在用户列表中，查看是否已存在刚刚添加的用户，

如果有，则说明页面及事件代码设置正确。如果页面有报错，请根据页面报错定位问题。

----结束

2.8.4 创建业务凭证

业务权限凭证是业务用户在登录AstroZero开发的应用后，所拥有的访问后端业务接口的凭证，后端业务接口以CustomAPI的形式进行开放时，可指定归属业务权限凭证。根据本示例的业务场景，创建4个业务凭证，即“cs”客服人员、“ds”派单人员、“ms”维修人员、“Login”业务用户登录专用。

操作步骤

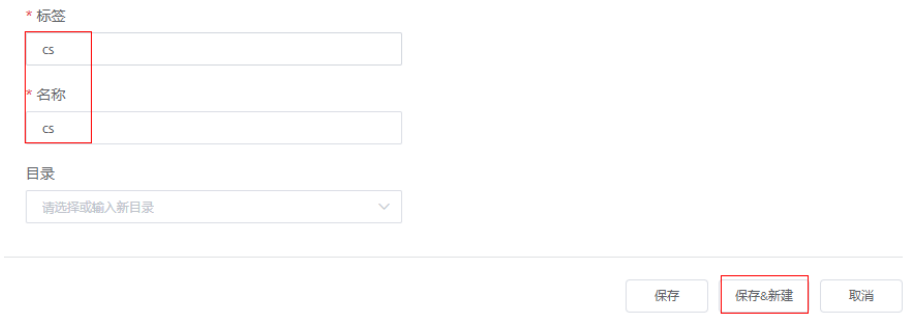
步骤1 使用华为账号，登录AstroZero经典版开发环境。

步骤2 在AstroZero经典版开发环境首页，单击“管理”，进入AstroZero经典版开发环境管理中心。

步骤3 选择“用户管理 >业务权限凭证”，在右侧“业务权限凭证”中，单击“新建”。

步骤4 在弹窗中，设置“标签”、“名称”为“cs”，单击“保存&新建”。

图 2-334 创建业务凭证。



步骤5 分别创建业务凭证“ds”、“ms”、“Login”。

----结束

2.8.5 创建权限配置

业务用户登录应用后的访问权限，是通过AstroZero的Profile进行控制管理的。

背景信息

本示例中创建了3种业务用户的业务权限，主要是在AstroZero预置的Portal User Profile权限基础上，进行自定义业务用户权限配置和拓展实现的。在AstroZero的权限配置功能中，基于某个权限配置的新创建的Profile，将会继承原Profile的全部权限。

在后续有新的业务用户注册时，只需要将业务用户添加到对应的权限配置中，即可获取该权限配置中的权限。

权限配置创建的大致流程：

1. 基于Portal User Profile创建客服人员csProfile。
2. 配置客服人员csProfile工作流权限、自定义对象权限以及业务凭证权限。
3. 在客服人员csProfile的基础上，创建派单人员dsProfile、维修人员msProfile，并修改对应的业务凭证权限。
4. 配置匿名用户登录权限配置的业务凭证。

操作步骤

步骤1 使用华为账号，登录AstroZero经典版开发环境。

步骤2 在AstroZero经典版开发环境首页，单击“管理”，进入AstroZero经典版开发环境管理中心。

步骤3 修改内置系统参数“bingo.permission.resource.default.switch”默认值。

1. 在左侧导航栏中，选择“系统配置 > 系统参数 > 内置系统参数”。
2. 在搜索框中，搜索“bingo.permission.resource.default.switch”，然后单击参数名。

图 2-335 修改内置系统参数默认值开发接口权限

3. 单击“值”右侧的，进入编辑页面，设置“值”为“是”，单击“保存”，返回参数详情页面。

图 2-336 编辑“值”

图 2-337 将值修改为“是”

修改当前参数是为了开启“Portal User Profile”权限配置中接口权限。

步骤4 创建“客服人员”的权限配置“csProfile”。

1. 选择“用户管理 > 权限配置”，在右侧“权限配置列表”中，单击“新建”。

图 2-338 新建权限配置

2. 在新建权限配置弹窗中，选择“现有权限配置”为“Portal User Profile”，设置要新增的“权限配置名称”为“csProfile”，单击“保存”，返回权限配置列表。

图 2-339 基于 Portal User Profile 新增权限配置

说明

- 普通克隆：权限配置创建后，不仅继承了原模板权限配置的权限，且后续可以根据实际业务在权限配置详情页进行修改。
- 继承克隆：除基本信息和业务权限凭证，其余权限屏蔽编辑按钮，不能修改。

3. 在权限配置列表中，单击“csProfile”，进入权限配置详情。
4. 配置“应用程序设置”下的权限。
   a. 在搜索框中搜索“设备维修管理系统”，设置当前应用“可见性”及“默认”。

   图 2-340 设置应用可见性及默认

   b. 单击“设备维修管理系统”，进入菜单可见性设置。
   c. 单击右上角，设置客服人员的默认菜单“工单列表（客服人员）”、及其他可见菜单，例如“Home”、“工单管理”，然后单击，返回应用列表，再单击，保存设置。

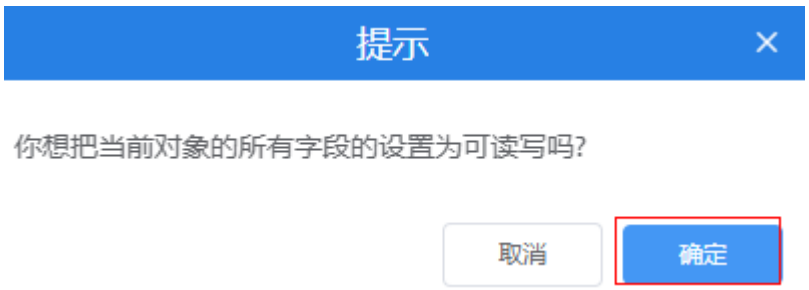
图 2-341 设置默认菜单为“工单列表（客服人员）”

5. 配置“自定义对象权限”下的对象权限。

   a. 在“自定义对象权限”页签，单击“自定义对象权限”右侧的，并在搜索框中输入“***HW***__WorkOrder__CST”，然后勾选“***HW***__WorkOrder__CST”对象的“编辑”权限，并在提示弹窗中单击“确定”。“***HW***__”请以实际命名空间前缀为准。

图 2-342 勾选自定义对象“编辑”权限

图 2-343 设置自定义对象所有字段可读写



b. 勾选“***HW***__WorkOrder__CST”对象的“修改全部”、“创建”权限，再单击，保存设置。

图 2-344 勾选“修改全部”

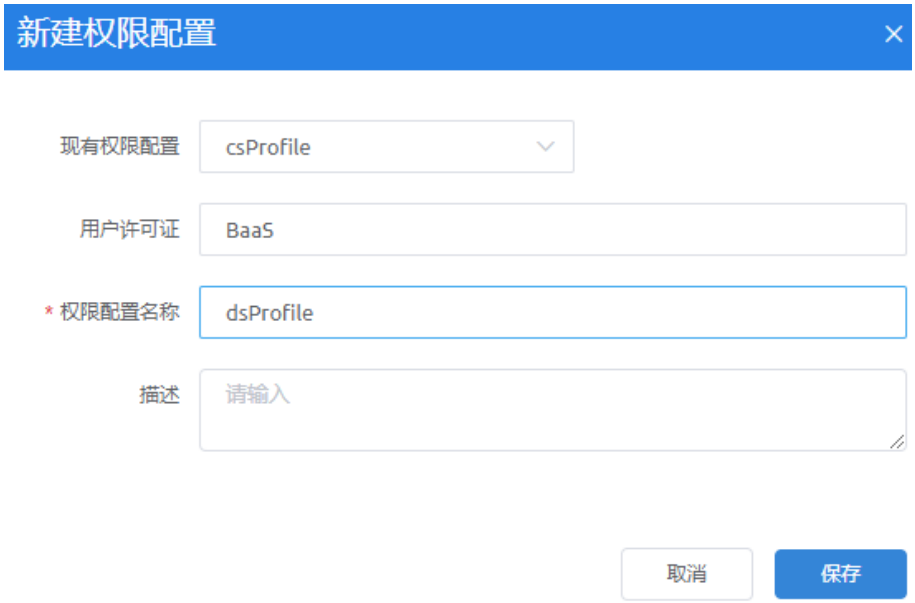
6. 单击“业务权限凭证”，在“业务权限凭证”下，勾选“cs”业务凭证，再单击，保存设置。

图 2-345 配置业务凭证

步骤5 创建“派单人员”的权限配置“dsProfile”。

1. 选择“用户管理 > 权限配置”，在右侧“权限配置列表”中，单击“新建”。
2. 在新建权限配置弹窗中，选择“现有权限配置”为“csProfile”，设置要新增的“权限配置名称”为“dsProfile”，单击“保存”，返回权限配置列表。

图 2-346 基于 csProfile 新增权限配置



3. 在权限配置列表中，单击“dsProfile”，进入权限配置详情。单击“应用程序设置”，单击“设备维修管理系统”，在弹窗中设置派单人员的默认菜单，即可见菜单如图2-347所示。

图 2-347 设置派单人员默认菜单

4. 单击“业务权限凭证”，在“业务权限凭证”下，取消勾选“cs”，勾选“ds”业务凭证，再单击✓，保存设置。

图 2-348 修改业务权限凭证

步骤6 创建“维修人员”的权限配置“msProfile”。

1. 选择“用户管理 > 权限配置”，在右侧“权限配置列表”中，单击“新建”。
2. 在新建权限配置弹窗中，选择“现有权限配置”为“csProfile”，设置要新增的“权限配置名称”为“msProfile”，单击“保存”，返回权限配置列表。
3. 在权限配置列表中，单击“msProfile”，进入权限配置详情。在“应用程序设置”页签，单击“设备维修管理系统”，在弹窗中设置维修人员的默认菜单，即可见菜单如图2-349所示。

图 2-349 设置维修人员默认菜单

4. 单击“业务权限凭证”页签，在“业务权限凭证”下，取消勾选“cs”，勾选“ms”业务凭证，再单击✓，保存设置。

图 2-350 修改业务权限凭证

步骤7 配置“Anonymous User Profile”的业务凭证。

1. 选择“用户管理 > 权限配置”，在右侧“权限配置列表”中。
2. 在权限配置列表中，单击“Anonymous User Profile”，进入权限配置详情。

图 2-351 单击 Anonymous User Profile

3. 单击“业务权限凭证”，在“业务权限凭证”下，勾选“Login”业务凭证，再单击✓，保存设置。

图 2-352 添加业务权限凭证

----结束

2.8.6 添加接口级业务权限凭证

业务用户的权限配置创建完成后，还需要为应用中创建的公共接口，添加接口级权限配置，本示例中业务用户需要使用的公共接口，主要是在“工单管理”及“用户管理”2个功能模块创建的。

操作步骤

步骤1 在“我的应用”中，单击“设备维修管理系统”，进入应用。

步骤2 在应用中，单击下方“配置”，进入应用配置。

图 2-353 应用配置入口

步骤3 在应用配置页面，单击“业务权限凭证”页签，进入业务权限凭证。

图 2-354 业务权限凭证

步骤4 单击“导入”，在弹窗中勾选“cs”、“ds”、“ms”、“Login”，然后单击“导入”。

图 2-355 导入业务权限凭证

步骤5 在页面下方，单击“服务”，进入公共接口页面。

图 2-356 创建公共接口入口

步骤6 为“createWorkOrder”公共接口，添加客服人员业务凭证“cs”。

1. 在接口列表中，单击“createWorkOrder”接口，进入接口详情。
2. 在接口详情页面，单击“业务权限凭证”下的“编辑”。

图 2-357 编辑业务权限凭证

3. 在编辑业务权限凭证弹窗中，勾选“cs”，再单击  ，然后单击“保存”。

图 2-358 选择客服人员业务凭证 cs

步骤7 参考**步骤6**，给其他公共接口添加业务权限凭证，详细接口名及业务凭证如**表2-50**所示。

表 2-50 需要添加业务凭证的公共接口

编号	公共接口名	需要添加的业务凭证	涉及业务用户
1	createWorkOrder（**上一步已添加**）	cs	客服人员
2	equipmentSelectListQuery	cs	客服人员
3	judgeNextStatus	ms	维修人员
4	modifyOrderStatus	ms	维修人员
5	queryWorkOrder	cs、ds、ms	客服人员、派单人员、维修人员

编号	公共接口名	需要添加的业务凭证	涉及业务用户
6	queryWorker	ds	派单人员
7	login	Login	客服人员、派单人员、维修人员

----结束

2.9 应用业务测试

2.9.1 本节导读

应用开发完成后，需要根据业务流程，通过使用不同角色账号登录，进行应用业务测试。

业务测试主要分为**管理业务用户**、**管理设备信息**和**处理工单**三部分。其中，前两部分是管理员权限进行的操作（这里管理员账号，均使用应用开发者账号进行）。处理工单部分，需要根据工单流程顺序，切换客服人员、派单人员、维修人员以及管理员号进行测试。

- **管理员用户**：当前登录AstroZero开发应用使用的账号，用于新增业务用户、添加用户权限、添加设备信息、管理工单，监控设备。
- **业务用户**：使用“设备维修管理系统”应用的用户，分别是客服人员、派单员及维修人员。

表 2-51 业务测试详情

角色		测试页面	涉及功能
管理员用户		新增用户页面、平台管理用户页面、设备管理页面和工单管理页面	新增业务用户，配置权限集、添加设备信息、管理工单
业务用户	客服人员	App登录页面，工单列表（客服人员）页面	创建工单
	派单员	App登录页面，工单列表（派单员）页面	派单、查看工单状态
	维修人员	App登录页面，工单列表（维修人员）页面	接单、拒单、关单

2.9.2 管理业务用户

管理业务用户模块主要包括新增业务用户、配置业务用户权限集两大功能。

- 新增业务用户

管理员用户登录应用预览页面后，在“新增用户”页面，新增业务用户。新增后，管理员用户可以在用户列表中，查看或删除用户。

- 配置业务用户权限集

  业务用户添加完成后，按照各自的角色功能，对每个业务用户添加业务凭证、设置权限配置，并在对应的工作队列中，添加业务用户。

  配置权限集后，业务用户将获得BPM、服务编排、自定义对象的操作和查看权限。

操作步骤

步骤1 使用应用开发者账号，访问并登录AstroZero经典版开发环境。

步骤2 在“项目 > 我的应用”中，单击“设备维修管理系统”应用的，进入应用预览页面。

图 2-359 查看应用

步骤3 新增业务用户。

1. 在应用菜单中，选择“新增用户”，进入用户管理页面。
2. 单击“新增用户”，在弹窗中输入用户名及密码，并设置当前新增业务用户的“角色”，单击“确定”。

图 2-360 新增用户

表 2-52 角色对应权限集

用户名	角色	对应权限集
testcs	客服人员	csProfile
testds	派单员	dsProfile
testms	维修人员	msProfile

步骤4 完成新增后，返回用户管理页面，查看是否已存在刚刚提交的用户信息。

图 2-361 查看用户

步骤5 增加业务用户的权限集。

1. 在浏览器中打开新的页签，使用当前应用开发者账号访问并登录AstroZero经典版开发环境。
2. 在经典版开发环境首页，单击“管理”，进入AstroZero经典版开发环境管理中心。

图 2-362 管理页签

3. 在左侧导航栏中，选择“用户管理 > 业务用户”，进入业务用户列表。
4. 在用户列表中，单击一个需要添加权限集的用户名，进入该用户信息详情。

图 2-363 选择业务用户名

5. 单击“覆盖业务用户权限”后的，勾选后，单击“保存”。

6. 单击“权限集”下的“编辑”，在弹窗中设置对应权限集，然后单击“保存”，返回用户详情查看已添加的权限集。

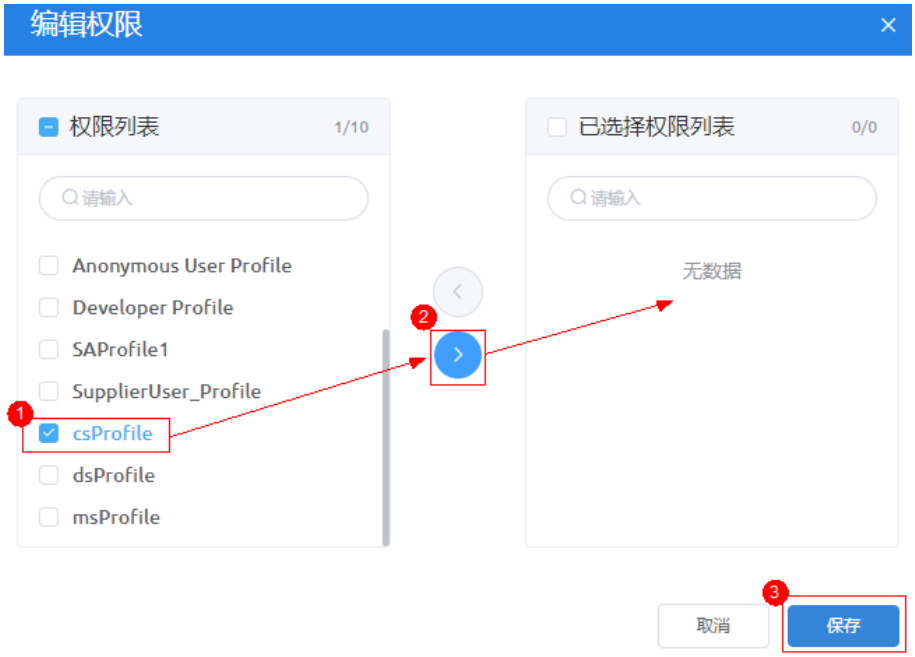
不同角色对应的权限集，如表2-53所示。

表 2-53 角色对应权限集

角色	对应权限集
客服人员	csProfile
派单员	dsProfile
维修人员	msProfile

图 2-364 业务用户详情

图 2-365 编辑业务用户权限列表



步骤6 在工作队列中，添加业务用户。

1. 在经典版开发环境首页，单击“管理”，进入经典版管理中心。
2. 在左侧导航栏中，选择“用户 > 工作队列”，单击客服人员队列“CustomerService”，进入队列详情。

3. 在队列详情中，单击“成员信息”下的“添加”，在弹窗中设置“成员类型”为“业务用户”，在“成员列表”下拉框中选中需要添加为客服人员的业务用户账号，单击“添加”。

图 2-366 在队列中添加业务用户

----结束

2.9.3 管理设备信息

应用开发完成后，设备信息中仅存在一些在创建过程中的测试数据，需要管理员对设备信息进行新增、编辑、删除等操作。

操作步骤

步骤1 使用应用开发者账号，访问并登录AstroZero经典版开发环境。

步骤2 在“项目 > 我的应用”中，单击“设备维修管理系统”应用的，进入应用预览页面。

步骤3 在应用菜单中，选择“设备管理”，在设备管理页面中，单击“新增设备”，在“设备详情”页，设置需要新增的设备信息，然后单击“保存”。

图 2-367 设备管理

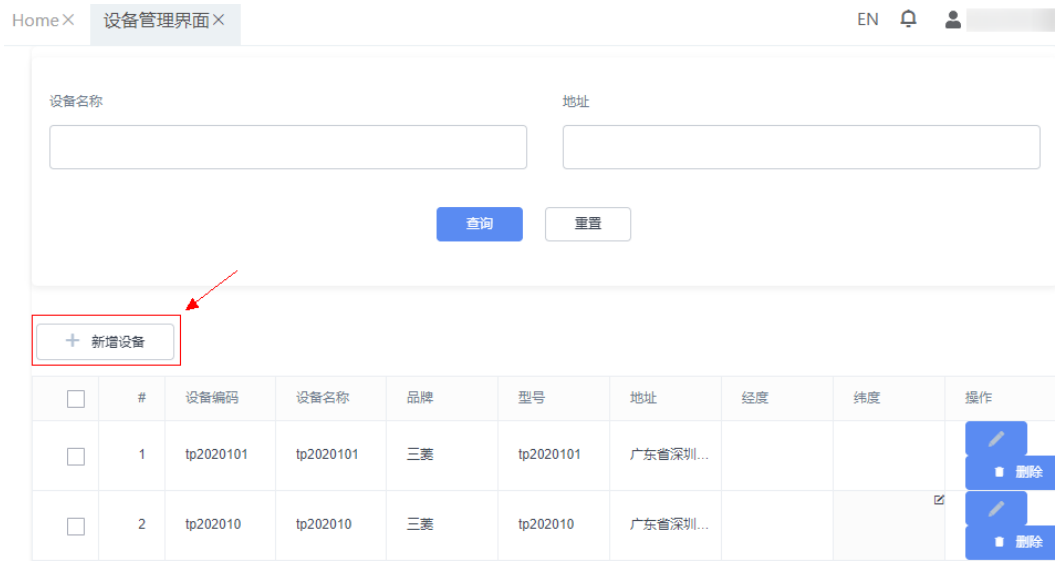


图 2-368 设置设备详细

步骤4 设备保存后，返回到设备管理页面，查看设备列表中是否已存在上一步添加的设备。

步骤5 选择一条设备记录，单击“操作”中的“编辑”及“删除”按钮，验证当前信息是否可以编辑及删除。

----结束

2.9.4 处理工单

处理工单需要使用不同的业务用户账号，通过登录对应功能页面，对工单流转进行处理。工单流转的流程顺序如下：

1. 客服人员创单。
2. 派单员派单。
3. 维修人员接单。

进入应用登录页面

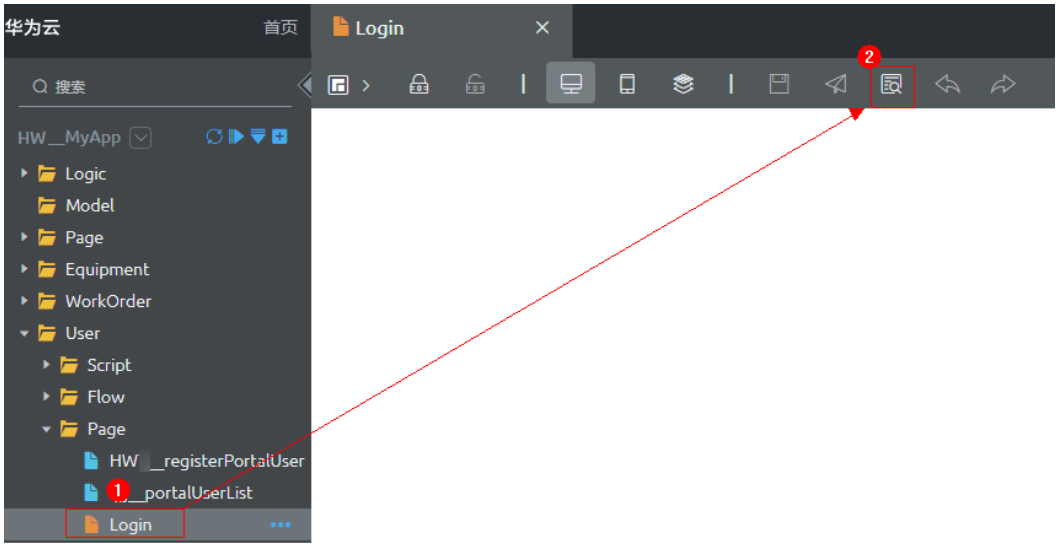
使用业务人员账号验证时，需要登录应用的登录页面。在开发环境中，您可以按照以下方式，进入应用登录页面。

步骤1 使用应用开发者账号，访问并登录AstroZero经典版开发环境。

步骤2 在“项目 > 我的应用”中，单击“设备运维管理”，进入应用。

步骤3 在应用菜单中的“User > Page”下，单击“Login”页面，再单击预览图标。

图 2-369 预览登录页



预览之后进入即可登录页，您可以在登录页使用对应的业务用户（例如testcs）登录。

图 2-370 进入 Login 页面的预览

说明

- 在开发环境中，业务用户需要在开发者账号登录的状态下，才能正常登录“设备运维管理”应用。
- 在开发环境中，使用业务用户账号在图2-370所示Login预览页，登录应用时，业务用户与当前预览页面的开发者账号共用同一个token，因此业务用户登录应用后，会导致开发者账号下线。当您要切换其他业务用户或返回开发环境时，需要重新使用开发者账号登录AstroZero开发环境。
- 如果在开发者账号未登录或已下线的状态下，直接使用业务用户账号登录应用，则属于匿名登录，如果出现以下错误属于正常现象。
  {"responseCode":401,"responseMessage":"Not allowed."},
- 当应用发布到运行环境后，业务用户则可以直接通过Login预览页URL登录应用，详细操作请参考在运行环境安装应用，发布应用到运行环境。

----结束

验证处理工单

步骤1 验证创建工单。

1. 访问图2-370中的**应用登录页**，使用客服人员账号登录（testcs）。
2. 登录后，进入“工单列表（客服人员）”页面，单击“创建工单”，进入创建工单页面。
3. 单击“选择工单方案”下拉框，设置“基本信息”区域的“设备名称”，检查“设备详情”区域显示了对应的设备详情。显示正确，则说明组件与模型的绑定，以及下拉框on-change事件执行正确。
4. 检查是否正常提交工单。

   填写工单的要素， 单击“提交”。如果页面跳转到工单列表页面，且在查询结果中，显示新创建的工单，则说明提交按钮的“点击”事件执行正确。

   如果页面跳转到工单列表页面，且在查询结果中显示新创建的工单，则说明验证成功。

图 2-371 验证结果样例

步骤2 验证派单功能。

1. 访问图2-370中的**应用登录页**，使用派单人员账号登录（testds）。
2. 登录后，进入“工单列表（派单员）”页面。

   正常情况下，系统会显示客服人员创建的工单，且每条记录后都有派单图标。

3. 选择上一步新建的一条工单记录，并单击该记录的 派发，弹出处理工单弹窗。

图 2-372 状态为“待派单”工单记录

4. 在处理工单弹窗中，设置下拉框“选择工程师”为“testms”，然后单击“确定”，返回“工单列表（派单员）”页面。

   查看工单记录的“状态”、“当前处理人”是否已更新为“待接单”、“testms”。如果已更新，说明派单流程正常。

图 2-373 处理工单-派单

说明

在派单页面进行派单时，派单状态不改变，并提示以下错误。可能是因为当前操作的工单信息是在“生成工单”（createWorkOrder）脚本中，使用测试数据创建的，脚本中测试数据创建的工单不支持在BPM状态流转。此时，请删除所有使用脚本创建的测试数据，使用创建工单页面（workOrderList）进行创建。

```
resCode: "405233002"
resMsg: "无法从请求的URL中获取正确的参数：'rootID'"
```

步骤3 验证待接工单。

1. 访问图2-370中的**应用登录页**，使用派单人员账号登录（testms）。
2. 登录后，进入“工单列表（维修人员）”页面。

   正常情况下，系统会显示派单员在上一步中派发的工单，如图2-374所示。

图 2-374 待处理工单

3. 单击“处理”，进入处理工单弹窗，“选择下一步操作”设置为“接单”，单击“提交”。

   处理完成后，返回“待处理工单”页面，查看“状态”是否已经更新为“处理中”。

   如果有多条工单，“待处理”状态的工单，优先显示，“处理中”的工单可能会显示在页面的下面，工单状态改变后，可以拖动滚动条查找该工单。

   说明

   如果在“选择下一步操作”中，选择了“拒单”，流程将返回“派单员”处，状态将变为“待派单”。

图 2-375 处理工单-关单

图 2-376 查看处理中的工单

4. 单击“处理中”工单的“处理”，选择“关单”，单击“提交”，返回页面后，查看该条工单的状态是否变成“关闭”，如果已关闭，则说明关单流程正常。

图 2-377 关闭工单

步骤4 管理工单。

1. 使用应用开发者账号，访问并登录AstroZero经典版开发环境。
2. 在“项目 > 我的应用”中，单击“设备维修管理系统”应用的，进入应用预览页面。

图 2-378 查看应用

3. 单击“工单管理”，进入工单管理页面，进行“创建工单”、派单及删除工单操作，验证工单管理功能。

----结束

2.10 打包发布

⚠ 注意

免费试用版本未开通运行环境权限，可能无法正常体验打包发布功能。

在应用开发完成后，应用需要编译、打包、发布，既可以发布到当前租户的运行环境，也可以共享给其他租户，在其他租户的开发环境或运行环境下安装。

发布须知

- **应用包编译类型**
  在AstroZero经典版应用设计器中，支持编译资产包和源码包两种的应用包。

  - 资产包：资产包中组件默认是设置为受保护状态，可以用在运行环境中安装后，进行预览，运行使用。也可以分享给其他租户安装，在其他租户的开发环境安装后，将显示在“库”页签下，资产包安装后仅能浏览不支持修改。
  - 源码包：源码包中的组件不受保护和限制。源码包可以分享给其他租户，其他租户安装在开发环境中后，可以进行二次开发，安装后将显示在“项目”页签下，详细操作请参见**在其他租户环境安装应用包**，另外，运行环境中不能安装源码包。

- **发布方式**

  AstroZero应用（资产包）发布有多种发布方式：我的仓库、Welink、华为OneMobile、微信、下载小程序，详细介绍请参考**发布App**。

  本示例中，主要介绍把应用发布到“我的仓库”，并通过“我的仓库”再部署到运行环境。

  说明

  编译类型为“源码包”时，只有发布到“我的仓库”一种发布方式。

编译与发布

步骤1 使用华为账号，访问并登录AstroZero经典版开发环境。

步骤2 在“项目 > 我的应用”中，单击“设备维修管理系统”，进入应用。

步骤3 单击左下角的  ，选择“设置”。

步骤4 在包类型中，选择“资产包”，单击“保存”。

图 2-379 设置包类型

步骤5 单击  ，选择“编译”，进行资产包编译。

如果想了解更多关于发布应用的设置，请参考**如何编译发布应用**。

步骤6 编译完成后，单击左下角  ，选择“我的仓库”，将应用程序安装包发布到当前租户的私有仓库。

步骤7 填写版本信息，单击“发布”。

发布成功后，页面显示“程序包已经被成功上传到我的仓库。”。

----结束

在运行环境安装应用

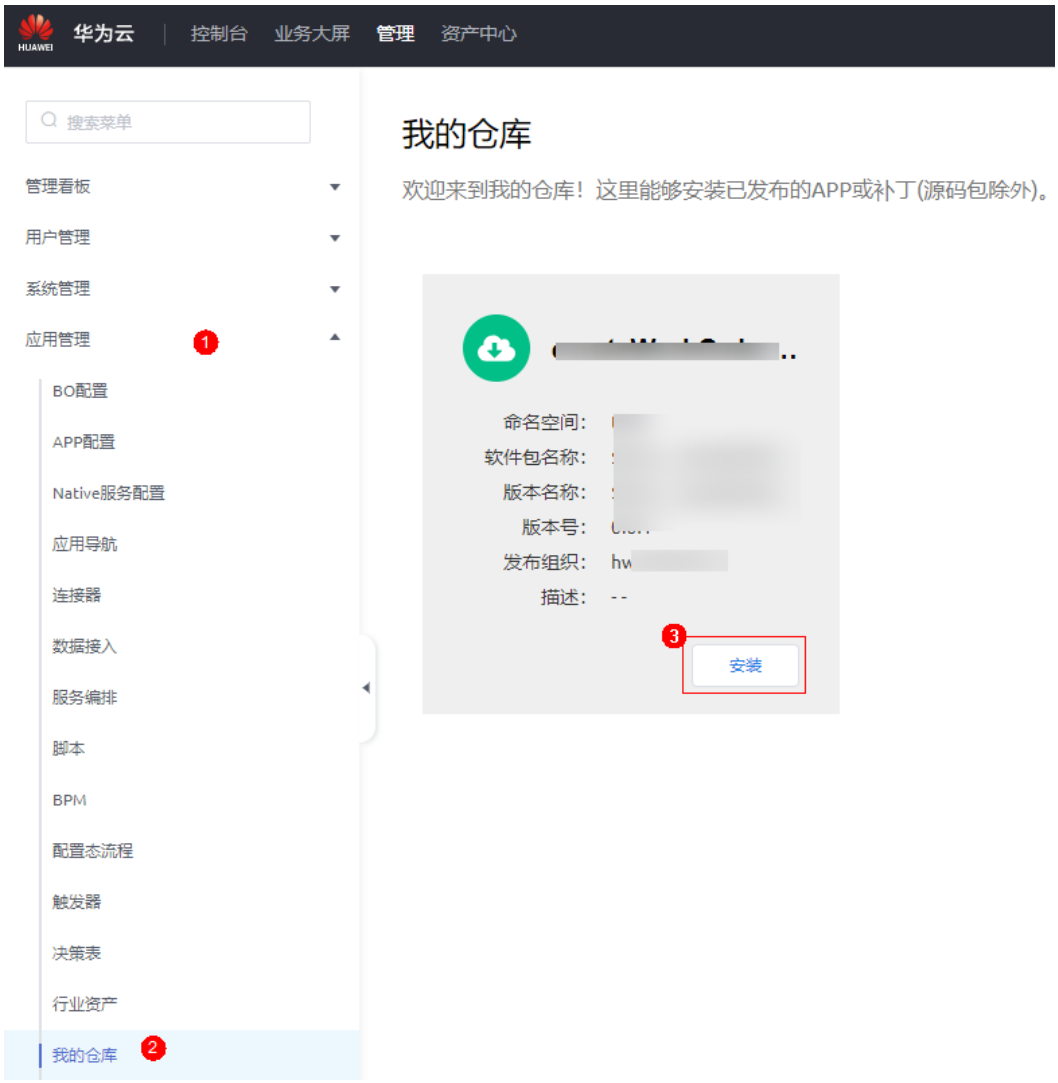
步骤1 使用开发者账号，访问并登录AstroZero经典版运行环境。

步骤2 在经典版运行环境首页，单击“我的仓库”或者在右侧菜单中，选择“应用管理 > 我的仓库”，在右侧中，找到**编译与发布**中发布的“设备维修管理系统”应用，单击“安装”。

说明

如果“我的仓库”下没有待安装的“设备维修管理系统”应用，您需要参考**编译与发布**中的操作方式，编译一个“资产包”应用，并选择发布到“我的仓库”。

图 2-380 在“我的仓库”安装设备运行管理应用



步骤3 预览应用。

1. 在左侧导航栏中，选择“应用管理 > 应用导航”。

2. 在右侧中，找到**编译与发布**中发布的“设备运维管理”应用，单击，进入应用布局页面，设置布局后，单击“保存”。
3. 单击，预览管理员的应用页面。

图 2-381 应用程序列表

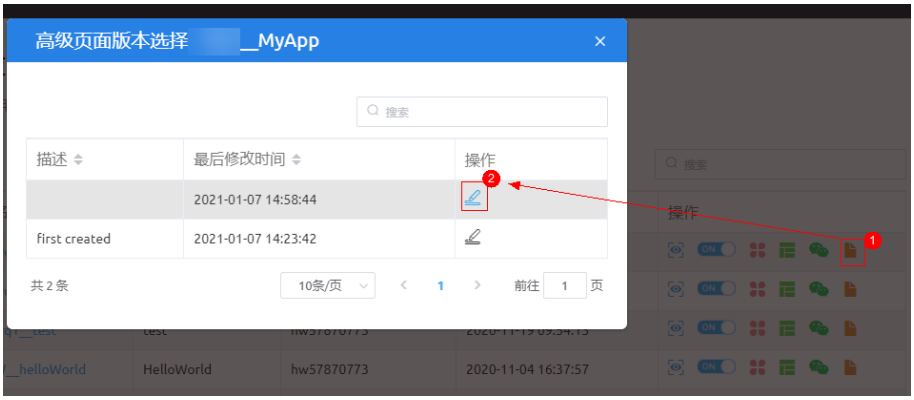
步骤4 （可选）获取登录页面URL，并分享登录页给应用业务用户。

说明

如果需要运行应用，则需要在运行环境“管理”中**创建工作队列**、**创建权限配置**，再参考**应用业务测试**中的步骤运行应用。

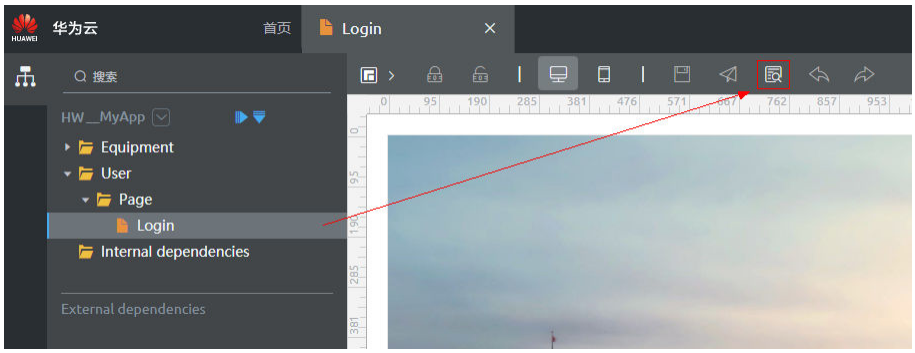
1. 在“应用程序列表”的“设备运维管理”应用中，单击，进入应用，查看应用的登录页面。

图 2-382 查看高级页面入口



2. 在“Page”中，单击“Login”页面，在页面中单击预览登录页面。

图 2-383 预览查看登录页 URL

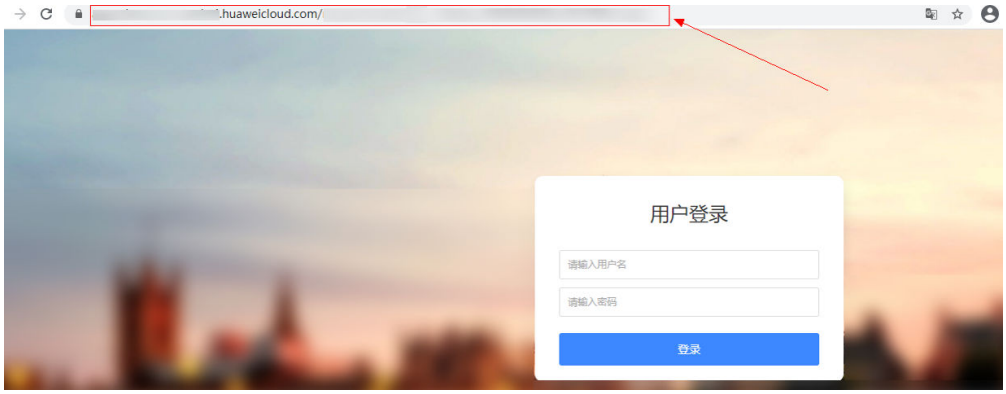


3. 预览之后进入登录页，复制登录页URL，并记录此地址。
   将登录页URL分享给业务用户，业务用户即可使用账号及密码进行登录。

   说明

   如果您还没有业务用户，您需要参考**管理业务用户**章节进行添加。

图 2-384 复制应用登录页 URL



----结束

在其他租户环境安装应用包

当其他租户需要安装此编译后的资产包或源码包时，请参考以下步骤进行安装。

步骤1 当前租户下载资产包或源码包。

在当前华为账号开发环境中，进入设备管理应用，在页面左侧单击  ，在“已发布”下，单击“资产包”或“源码包”，查看已发布的APP，在右侧页面单击“下载”。

步骤2 将下载后的应用包分享给其他租户。

步骤3 在其他华为账号下，以导入方式，安装应用包。

1. 使用华为账号，登录AstroZero经典版开发环境。
2. 在经典版开发环境首页，单击“管理”，进入AstroZero经典版开发环境管理中心。
3. 在左侧导航栏中，选择“应用管理 > 软件包管理 > 软件包安装”。
4. 单击“新建”，在“软件包安装”页面，拖入待安装的应用包，单击“安装”。

如果安装的是资产包，请在“库”页签下查看。如果安装的是源码包，请在“项目”页签下的“我的应用”中查看。

----结束

3 对象专项

3.1 使用 AstroZero 将客户与订单数据关联并同步修改

期望实现效果

在某些订单系统中，通常需要将客户信息和订单数据进行关联，用于处理订单、扣减库存等。例如，订单应用“A”中存在客户对象customerList和订单对象orderList两个对象，在AstroZero中通过为两个对象建立关联关系，可轻松实现如下功能：

- 客户对象页面可以新增、展示客户对象数据

  图 3-1 新增客户对象数据

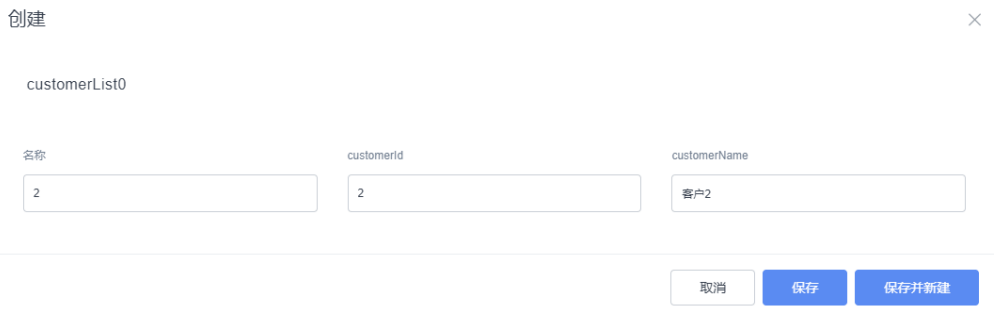
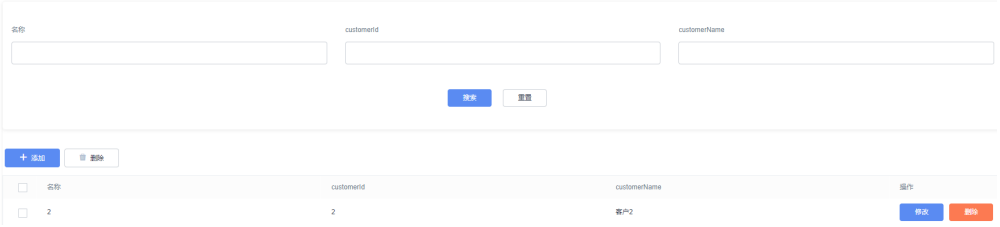
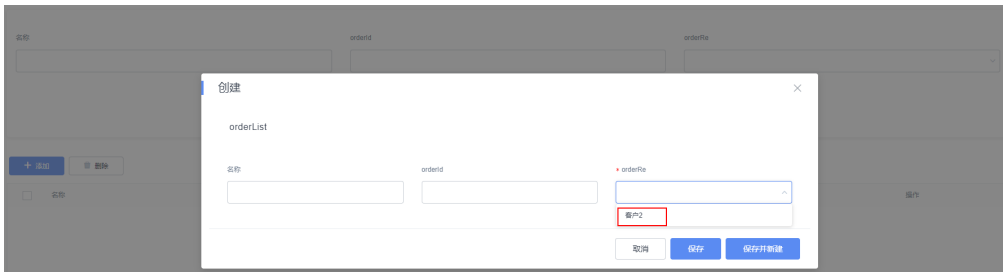
  

  图 3-2 展示客户对象数据

- 订单对象页面可以新增订单数据，关联已有客户信息

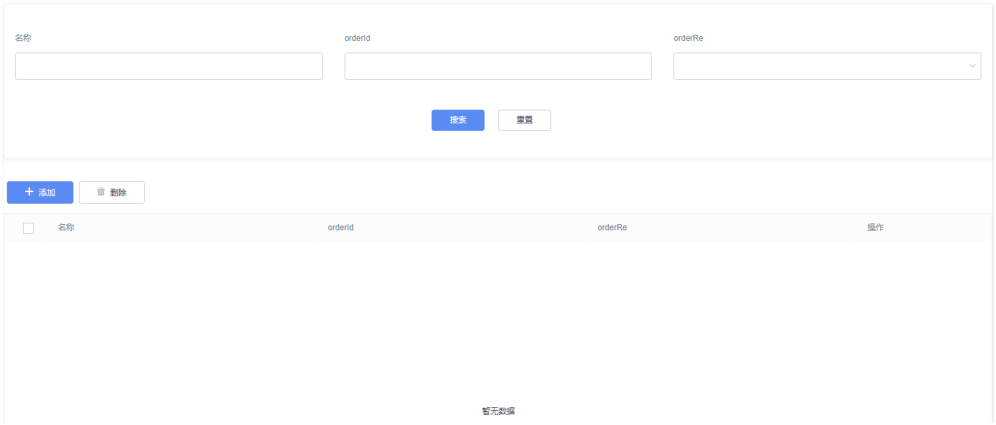
图 3-3 新增订单数据、关联已有客户信息



- 删除客户数据后，订单数据也随之删除

图 3-4 删除客户数据

图 3-5 订单数据同时被删除



功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 3-6 创建一个空白应用

表 3-1 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建客户对象customerList和订单对象orderList，并为对象添加字段。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识，单击“确定”。

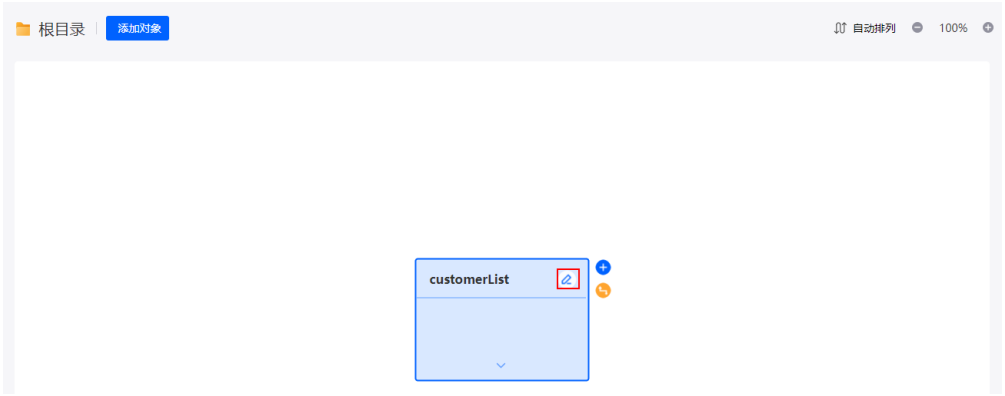
图 3-7 创建对象 customerList

表 3-2 新建 customerList 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	customerList
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	customerList

3. 在已创建的对象中，单击，进入对象详情页面。

图 3-8 选择编辑按钮



4. 在“字段”页签，单击“添加”，为对象添加customerId字段。

图 3-9 添加 customerId 字段

表 3-3 添加 customerId 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	customerId

参数	说明	示例
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	customerId
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本
数据长度	允许输入字段的长度。	64

5. 在“字段”页签，再次单击“添加”按钮，添加customerName字段。

图 3-10 添加 customerName 字段

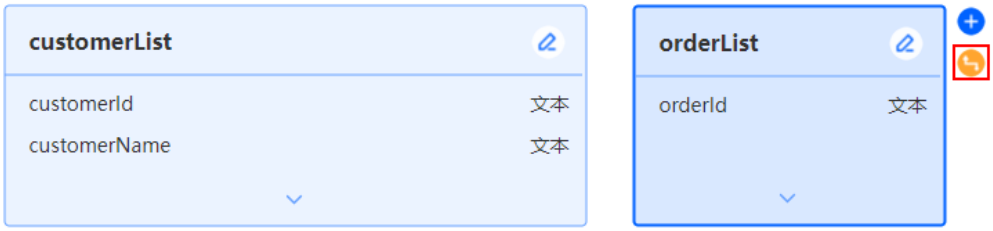
6. 按照上述操作，创建订单对象orderList，并为其添加orderId字段。

图 3-11 创建 orderList 对象并添加字段

步骤3 选择订单对象，添加关联。

1. 单击对象上的 ，进入添加关联页面。

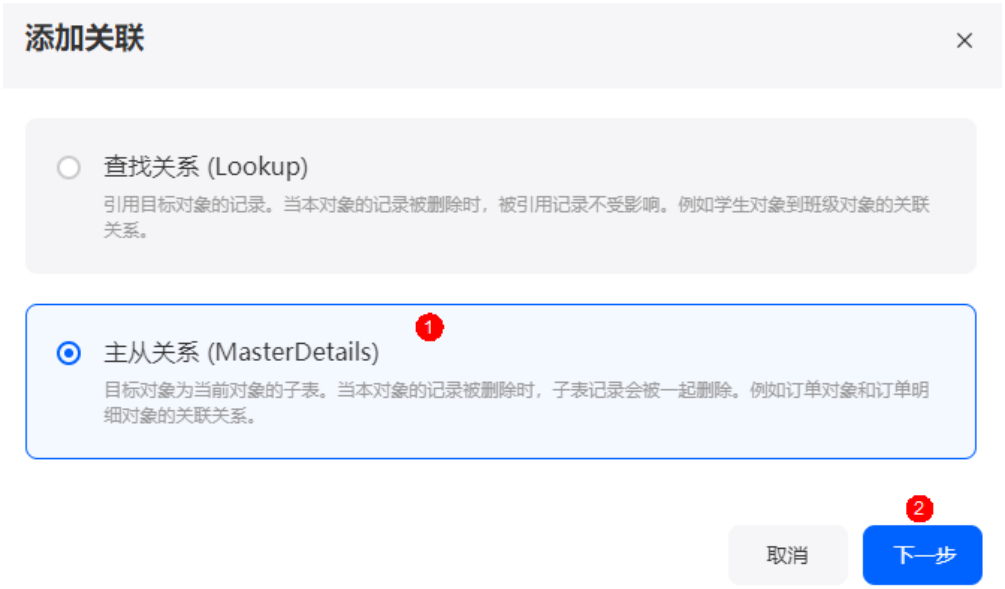
图 3-12 选择关联图标



2. 选择“主从关系”，单击“下一步”。

   主从关系是指目标对象为当前对象的子表，通过关联当前字段与另一对象的ID字段，创建本对象与另一对象的主从关系。定义了主从关系后，本字段的取值只能来源于关联主对象。当本对象的记录被删除时，子表记录会被一起删除。

图 3-13 选择主从关系



3. 添加关联关系，单击“确定”。

图 3-14 添加关联

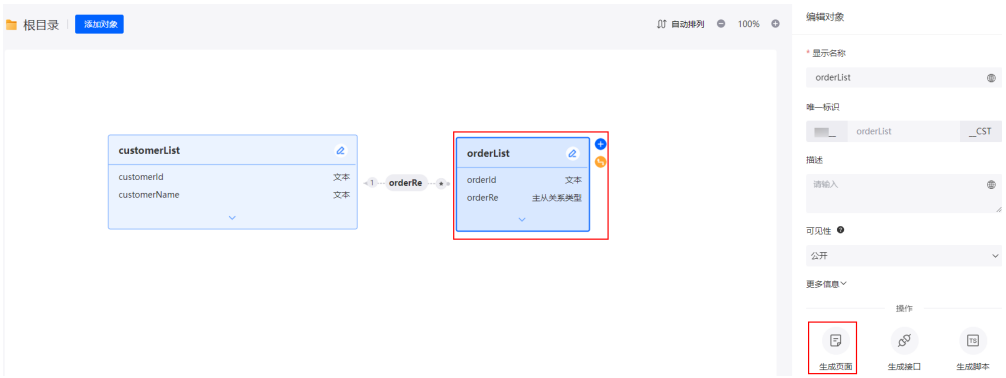
表 3-4 添加 orderRe 关联参数说明

参数	说明	示例
显示名称	关联关系在页面显示的名称，创建后可修改。 取值范围：1~80个字符。	orderRe
唯一标识	关联关系在系统中的唯一标识，创建后不可修改。 命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	orderRe
关联对象	选择关联目标，下拉框展示当前应用内所有对象的显示名称。	选择步骤2中创建的客户对象customerList。

步骤4 分别将客户和订单对象生成页面。

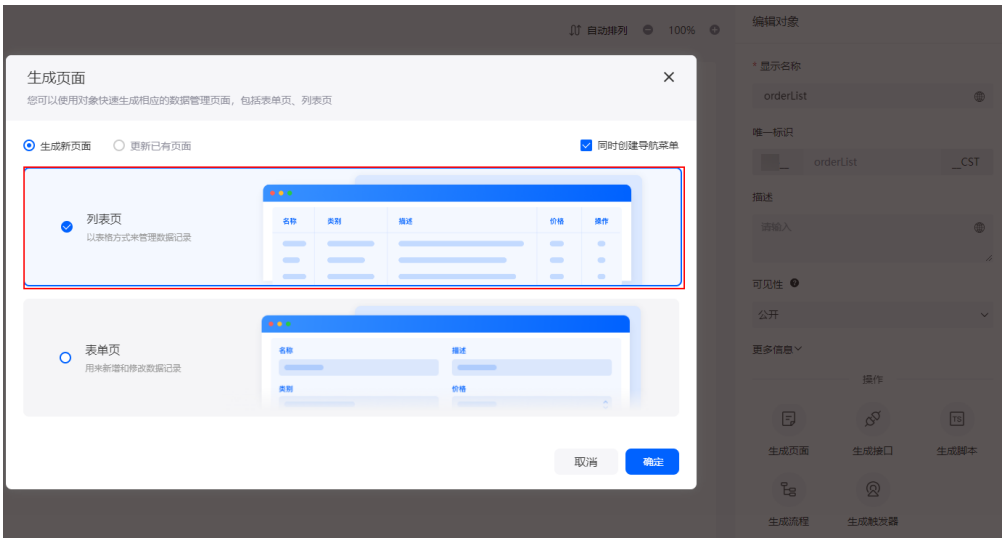
1. 选中订单对象orderList，在右侧编辑对象中，选择“操作 > 生成页面”。

图 3-15 选择生成页面



2. 取消选中“表单页”，仅保留“列表页”，单击“确定”。

图 3-16 将 orderList 对象生成列表页

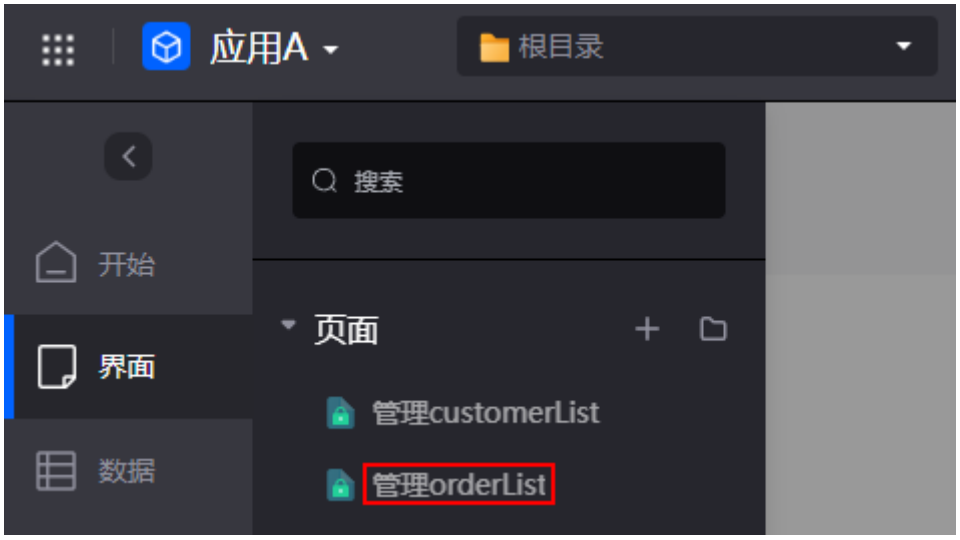


3. 按照同样的方法，将客户对象customerList也生成列表页。

步骤5 在订单对象页面设置父对象显示的字段。

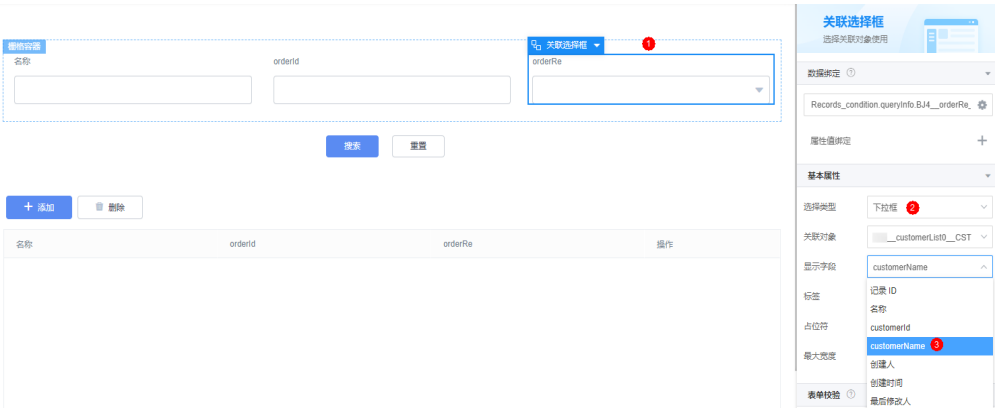
1. 在应用设计器的左侧导航栏中，选择“界面”。
2. 在“页面”中，单击“管理orderList”。

图 3-17 单击订单对象页面 orderList



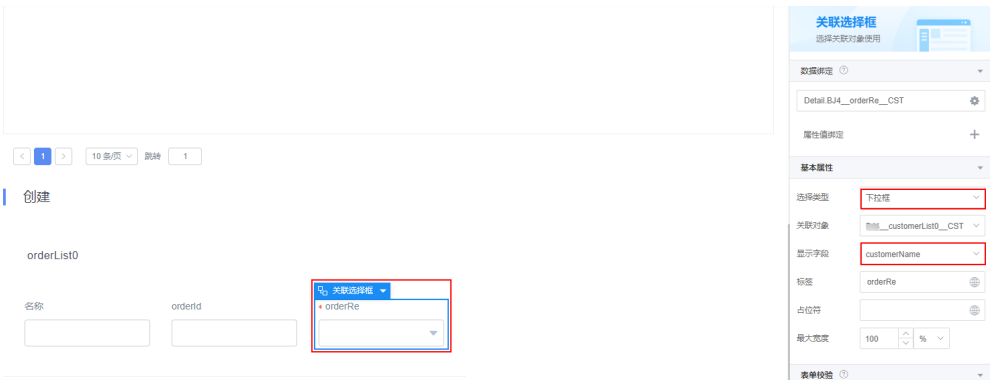
3. 选中“orderRe”字段，将“属性 > 基本属性”中的“选择类型”设置为“下拉框”，“显示字段”设置为“customerName”。

图 3-18 设置父对象显示字段



4. 在下方的创建中，同样选中“orderRe”字段，将“属性 > 基本属性”中的“选择类型”设置为“下拉框”，“显示字段”设置为“customerName”。

图 3-19 为创建页面设置父对象显示的字段



步骤6 单击页面上方的，保存页面。

步骤7 保存成功后，单击页面上方的，查看页面配置效果。

----结束

3.2 使用 AstroZero 在前端表格中增删改对象数据

期望实现效果

通过添加一个工具栏，在前端页面实现对象数据的增加、删除和修改。例如，在标准页面的表格中，增加、删除和修改数据时，页面关联的后台对象数据也会随之更改。

最终实现效果：双击demoName中的数据栏，可以对数据进行编辑。编辑后，单击“保存”按钮，可发现表格中数据已经被更新，同时后台对象也已被更新。选择对应的表格列，单击“删除”按钮，确认删除。删除成功后，后台对象中的数据也会随之删除。

图 3-20 在页面更新数据

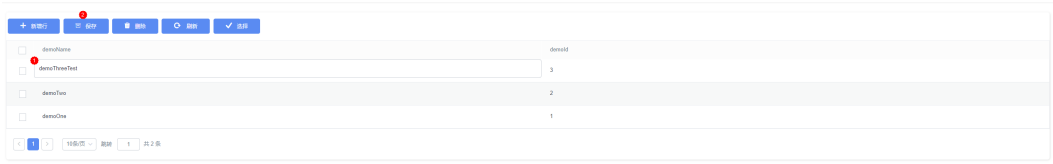


图 3-21 对象中数据也随之更新

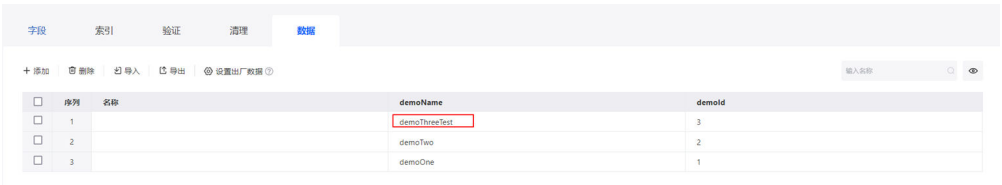


图 3-22 执行删除操作

图 3-23 页面中数据被删除

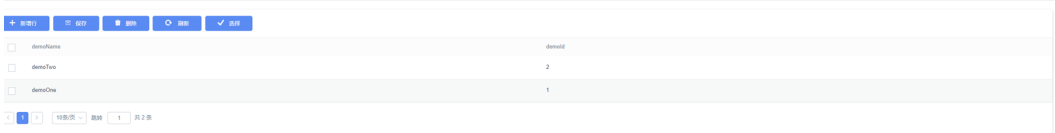


图 3-24 对象中数据也随之删除

功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 3-25 创建一个空白应用

表 3-5 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建对象“demoData”，并为对象添加字段和数据。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“demoData”，单击“确定”。

图 3-26 创建对象 demoData

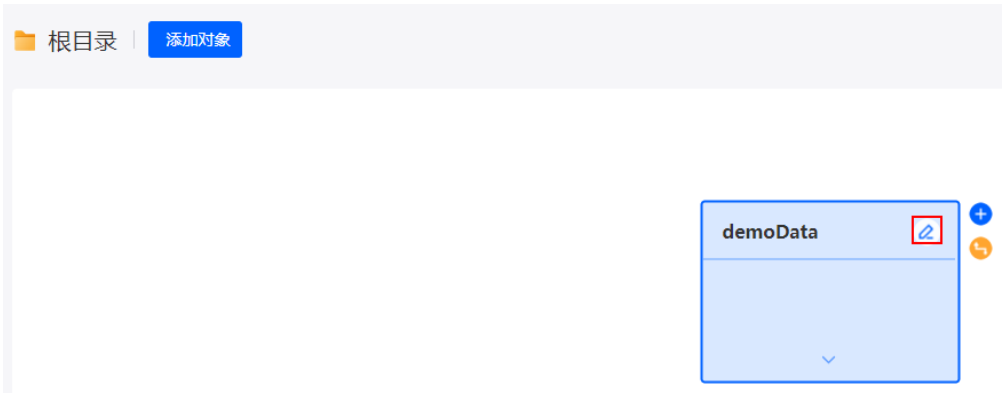
表 3-6 新建 demoData 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	demoData

参数	说明	示例
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	demoData

3. 在已创建的对象中，单击，进入对象详情页面。

图 3-27 选择编辑按钮



4. 在“字段”页签，单击“添加”，为对象添加demoName字段。

图 3-28 添加 demoName 字段

表 3-7 添加 demoName 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	demoName
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	demoName
字段类型	单击 ···，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

5. 在“字段”页签，再次单击“添加”按钮，添加demoId字段。

图 3-29 添加 demoId 字段

6. 选择“数据”页签，单击“添加”，为对象添加数据。

图 3-30 为对象添加数据

步骤3 新建对象模型。

1. 在应用设计器的左侧导航栏中，选择“界面”，单击页面后的“+”。
2. 输入页面的标签和名称，单击“添加”，新建一个标准页面。

图 3-31 新建一个标准页面

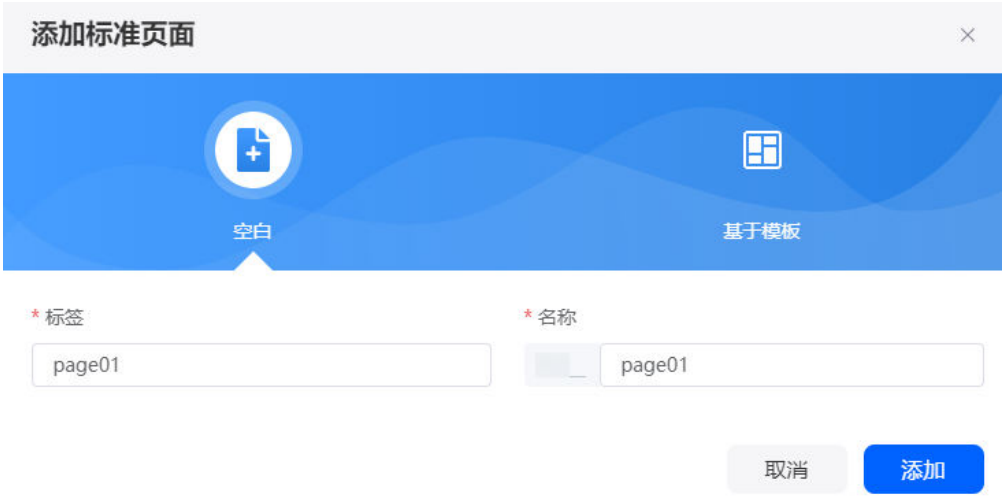


表 3-8 标准页面参数说明

参数	说明	示例
标签	标准页面的标签名，创建后可修改。 取值范围：1~64个字符。	page01
名称	标准页面的名称，名称是标准页面在系统中的唯一标识，创建后不可修改。命名要求如下： – 长度不能超过64个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	page01

3. 在标准页面的底部，单击“模型视图”，从设计视图切换到模型视图。

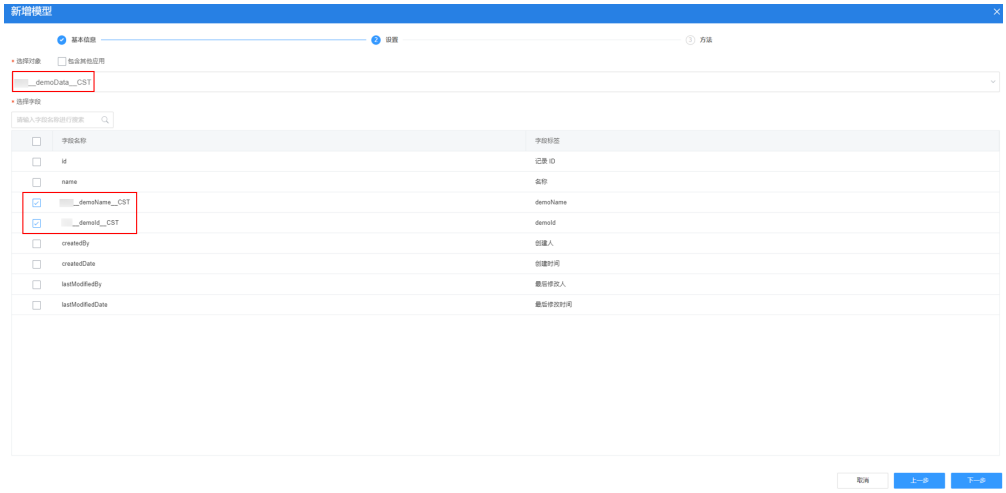
图 3-32 单击模型视图

4. 单击“新增模型”，输入模型名称（如demo）、“来源”选择“对象”，单击“下一步”。

图 3-33 新建模型

5. 选择**步骤3**中创建的对象和添加的字段，单击“下一步”。

图 3-34 选择对象和字段

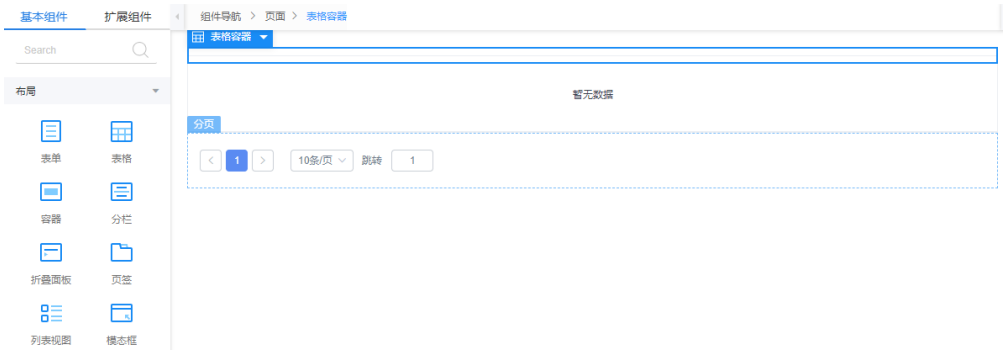


6. 单击“确定”，完成模型的创建。

步骤4 返回设计视图页面，新建表格关联模型。

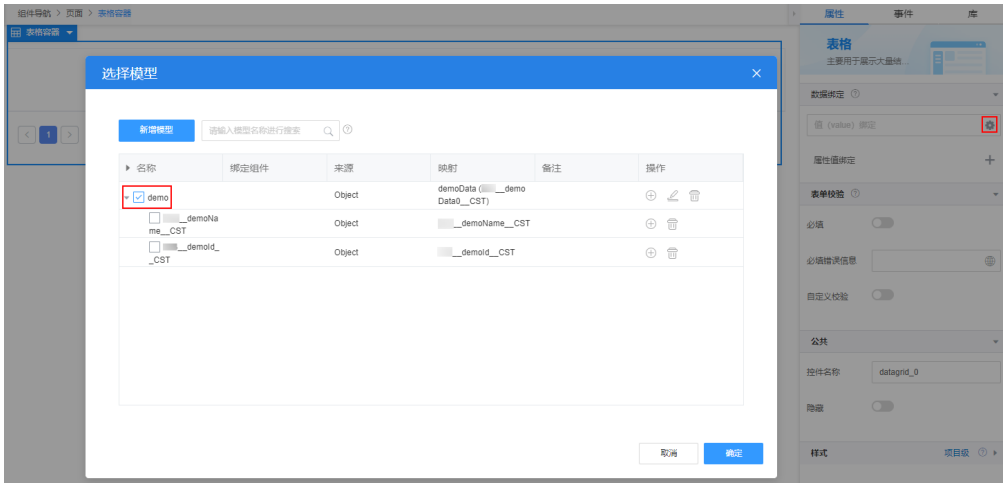
1. 在标准页面的底部，单击“设计视图”，从模型视图切换回设计视图。
2. 在标准页面中，拖入一个表格组件。

图 3-35 拖入表格组件



3. 选中表格组件，在“属性 > 数据绑定 > 值绑定”中，单击 。
4. 选中**步骤3**中创建的模型，单击“确定”。

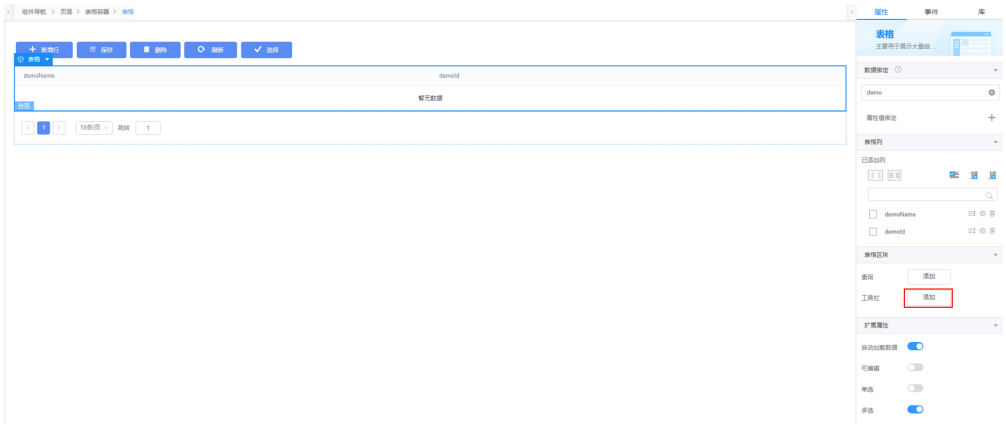
图 3-36 选择模型



步骤5 添加工具栏。

1. 选中表格组件，在“属性 > 表格区块”中，单击工具栏后的“添加”，添加一个工具栏。

图 3-37 添加工具栏



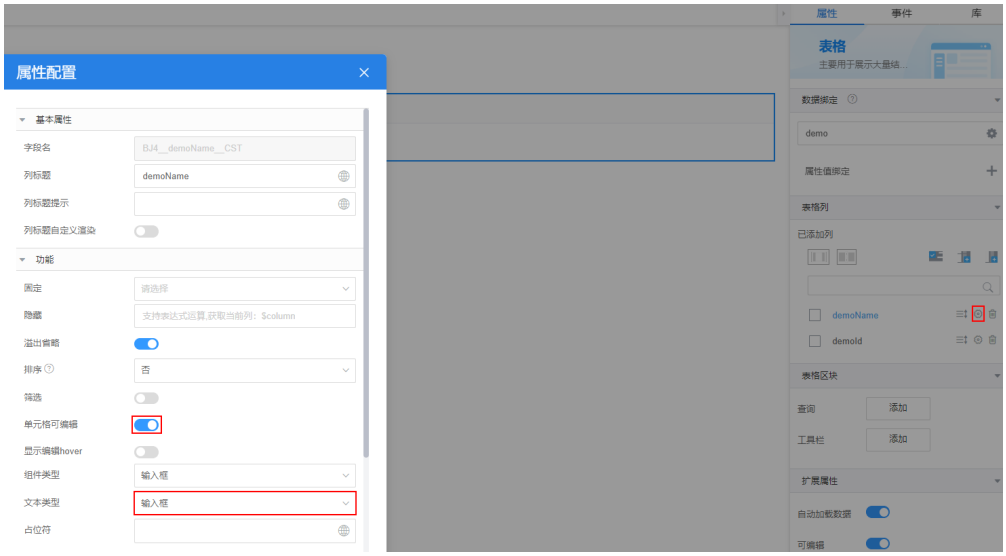
2. 在扩展属性中，开启“可编辑”。

图 3-38 开启可编辑

3. 将demoName、demoId列设为可编辑。

   在已添加列中，单击demoName后的 ，将列设置为可编辑。按照上述操作，将demoId列也设置为可编辑。

图 3-39 将列 demoName 设置为可编辑



步骤6 单击页面上方的，保存页面。

步骤7 保存成功后，单击页面上方的，查看页面配置效果。

----结束

4 脚本专项

4.1 通过 AstroZero 中的脚本实现表单的提交限制功能

期望实现效果

在开发前端页面时，可以在脚本中为表单添加一些提交限制，来提升用户体验和数据的安全。例如，在脚本中定义一个延迟时间（如图4-1），在规定的时间内提交表单时，提示“Submission failed: Not PortalUser!”（如图4-2）；超出规定的时间，则提示“Submission failed: Submitted too late”（如图4-3）。

图 4-1 通过脚本限定

```
try {
    let currentTime = now();
    let date = toDate('2024-04-08 20:08:08', 'yyyy-MM-dd HH:mm:ss');
    if (date.getTime() < currentTime.getTime()) { // 限制提交时间
        error.name = "WOERROR";
        error.message = "Submitted too late";
        throw error;
    }
```

图 4-2 非业务用户无法提交

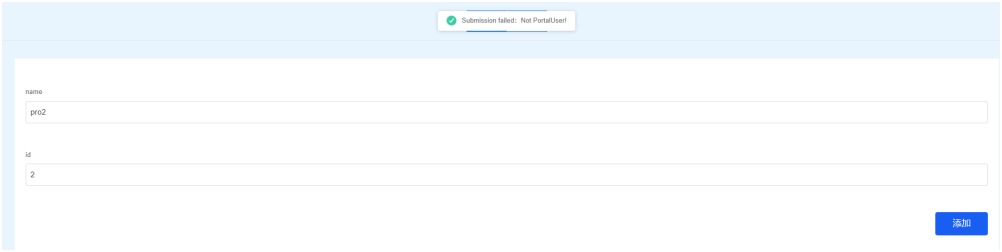
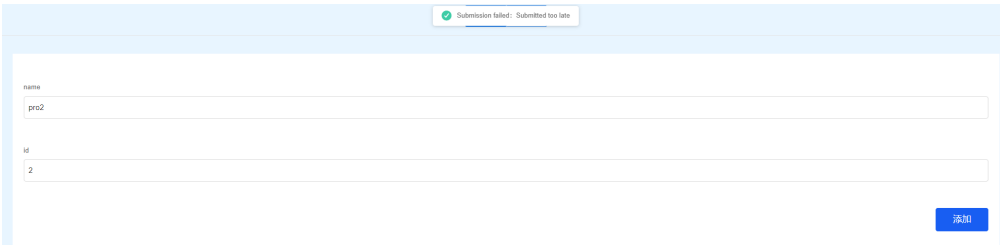


图 4-3 提示提交过晚



功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 4-4 创建一个空白应用

表 4-1 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建对象“product”，并为对象添加字段。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“product”，单击“确定”。

图 4-5 创建对象 product

表 4-2 新建 product 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	product

参数	说明	示例
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	product

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加ProName字段。

图 4-6 添加 ProName 字段

表 4-3 添加 proName 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	proName
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	proName
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

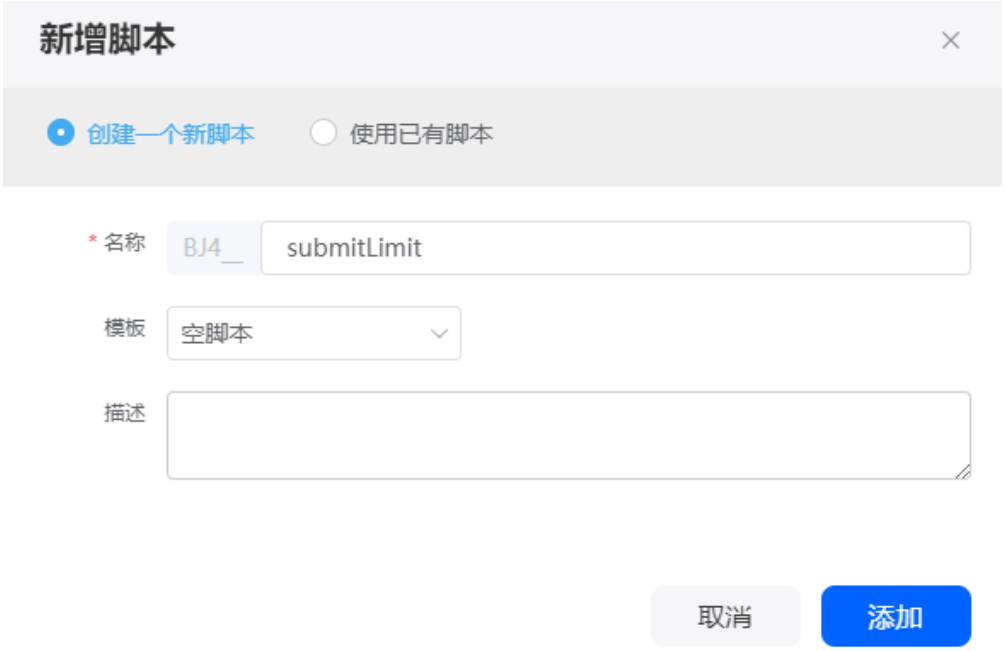
5. 在“字段”页签，再次单击“添加”按钮，添加proId字段。

图 4-7 添加 proId 字段

步骤3 创建一个脚本。

1. 在应用设计器中，选择“逻辑”，单击脚本后的“+”。
2. 新建一个空白的脚本，名称设置为“submitLimit”，单击“添加”。

图 4-8 创建脚本 submitLimit



3. 在脚本编辑器中，输入示例代码。

   本示例代码主要实现的功能为：获取前端页面输入，对入参进行校验，并调用对象接口完成对象实例增加。示例中的“命名空间__product__CST”为**步骤2**中创建的对象。

```
//本脚本用于提交表单以及限制提交时间、用户类型
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import { now } from 'date';
import { toDate } from 'date';

//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    proId: string;
    @action.param({ type: 'String', required: true, label: 'String' })
    proName: string;
}
//定义出参结构，出参包含1个参数，workOrder的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
}
//使用数据对象命名空间__product__CST
@useObject(['命名空间__product__CST'])
@action.object({ type: "method" })
export class CreateWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public createWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let currentTime = now();
            let date = toDate('2024-04-08 20:08:08', 'yyyy-MM-dd HH:mm:ss');
            if (date.getTime() < currentTime.getTime()) { // 限制提交时间
                error.name = "WOERROR";
                error.message = "Submitted too late";
                throw error;
            }
```

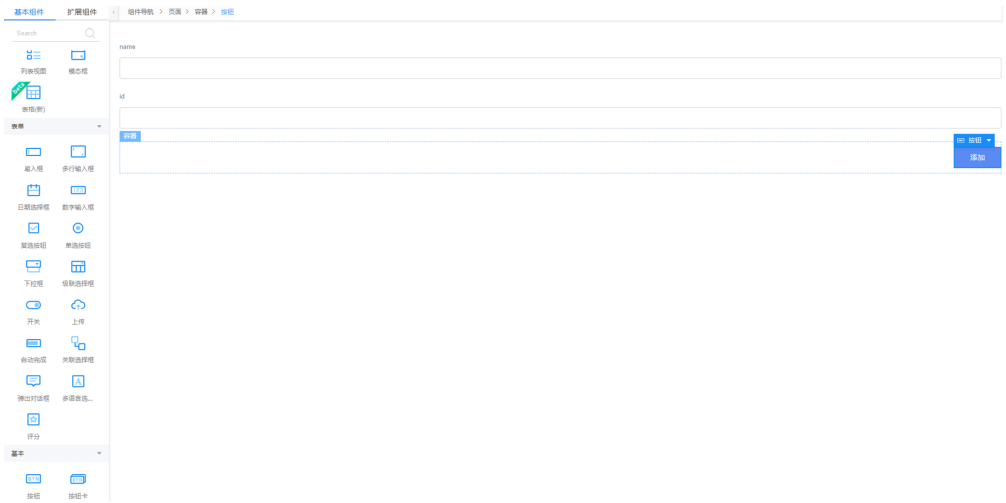
```
        let user = context.getUserType();
        if (user != "PortalUser") { // 限制提交用户类型，只有业务用户PortalUser可以提交
            error.name = "WOERROR";
            error.message = "Not PortalUser!";
            throw error;
        }

        let productData = new Object();
        productData['命名空间__proName__CST'] = input.proName;  //入参赋值
        productData['命名空间__proId__CST'] = input.proId;
        let s = db.object('命名空间__product__CST');  //获取命名空间__product__CST这个Object的操作
实例
        let id = s.insert(productData);
        if (id) {
            out.id = id;
        } else {
            error.name = "WOERROR";
            error.message = "Unable to create product!";
            throw error;
        }
    } catch (error) {
        console.error(error.name, error.message);
        context.setError(error.name, error.message);
    }
    return out;
  }
}
```

4. 单击  ，保存脚本，保存成功后单击  ，激活脚本。

步骤4 创建表单页面，用于提交表单数据。

1. 在应用设计器中，选择“界面”，单击页面后的“+”。
2. 输入页面的标签和名称，单击“添加”，新建一个标准页面。
3. 从基本组件中，拖拽2个输入框组件和1个按钮组件到右侧画布中。
4. 选中第一个输入框，将标签修改为“name”，将第二个输入框的标签修改为“id”，将按钮显示名称设置为“添加”。

图 4-9 设计表单页面



步骤5 新建对象模型。

1. 在标准页面的底部，单击“模型视图”，从设计视图切换到模型视图。
2. 单击“新增模型”，输入模型名称（如submitLimit）、“来源”选择“服务”，单击“下一步”。

图 4-10 新建模型

3. “选择服务类型”设置为“脚本”，在弹出的选择服务页面选择**步骤3**中创建的脚本，单击“确定”。

图 4-11 选择脚本

4. 单击“下一步”，再单击“确定”，完成模型的创建。

步骤6 返回设计视图页面，新建表格关联模型。

1. 在标准页面底部，单击“设计视图”，从模型视图切换回设计视图。
2. 选中name输入框，在“属性 > 数据绑定 > 值绑定”中，单击 。
3. 选择**步骤5**中创建的模型（proName），单击“确定”。

图 4-12 选择模型

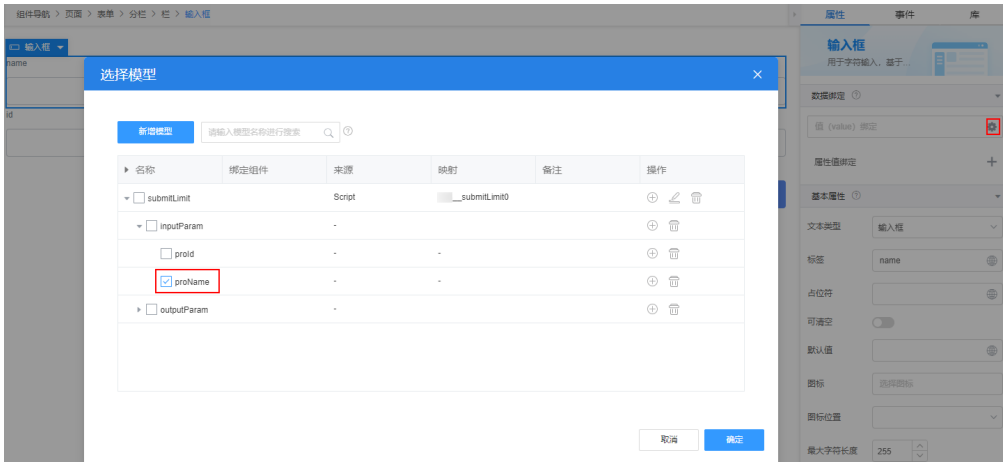
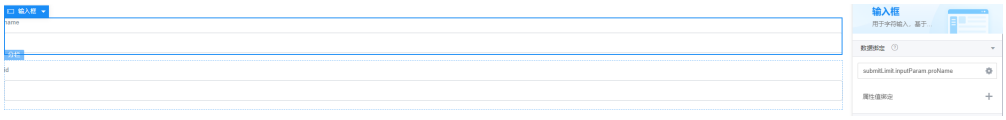
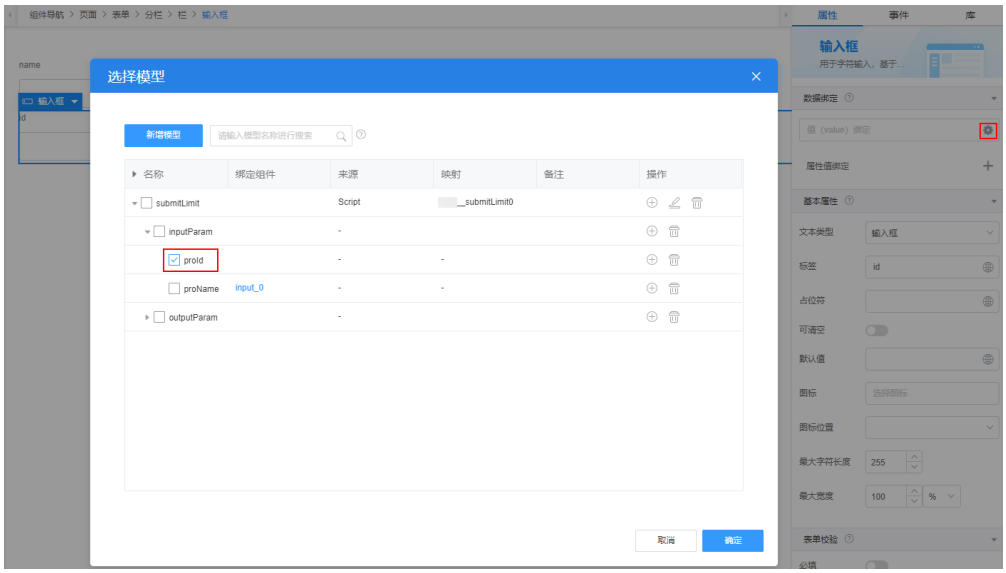


图 4-13 绑定后效果



4. 按照上述操作，为id输入框绑定**步骤5**中创建的模型（ proId ）。

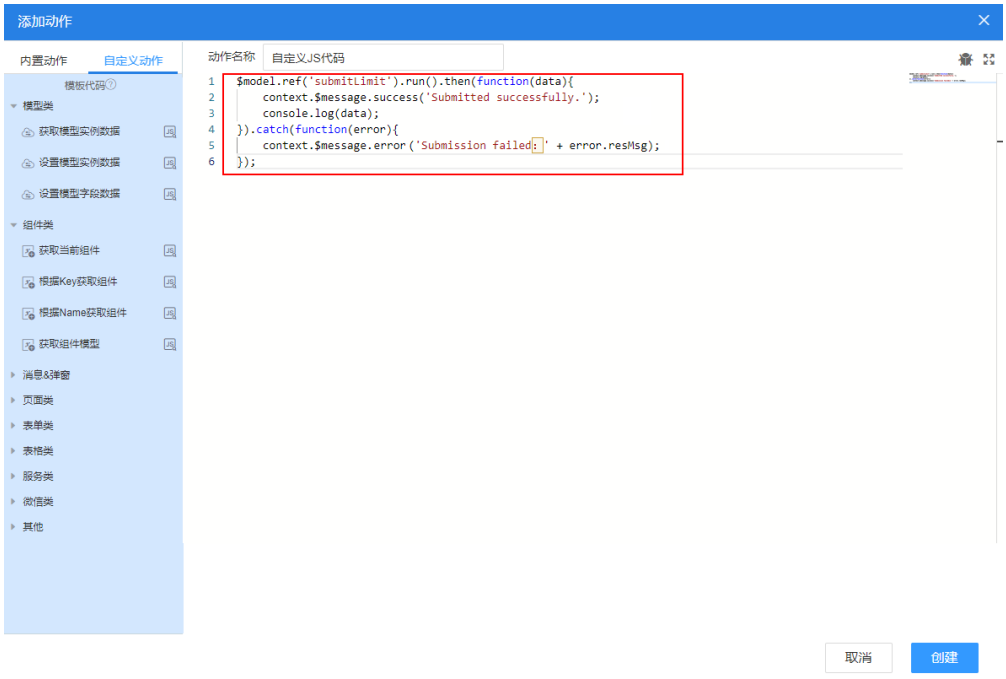
图 4-14 为 id 输入框绑定模型



步骤7 为添加按钮，添加事件。

1. 选中添加按钮组件，在页面右侧选择“事件”页签。
2. 单击“点击”后的“+”，进入添加动作页面。
3. 在自定义动作中，输入自定义代码，单击“创建”。

图 4-15 自定义动作



本示例中自定义的JS代码主要实现的功能为：单击提交按钮后，获取脚本抛出的异常信息并在页面展示。

```
$model.ref('submitLimit').run().then(function(data){
    context.$message.success('Submitted successfully.');
    console.log(data);
}).catch(function(error){
    context.$message.error('Submission failed：' + error.resMsg);
});
```

步骤8 单击页面上方的，保存页面。

步骤9 保存成功后，单击页面上方的，预览效果。

----结束

4.2 通过 AstroZero 中的脚本实现表格数据的增加和删除

期望实现效果

通过脚本，在前端页面实现对象数据的增加和删除。例如，在标准页面中增加或删除一条数据时，标准页面关联的对象中，数据也会随之添加或删除。

图 4-16 在页面添加数据

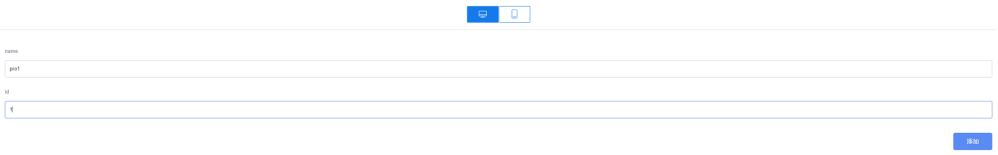


图 4-17 对象中同步新增数据

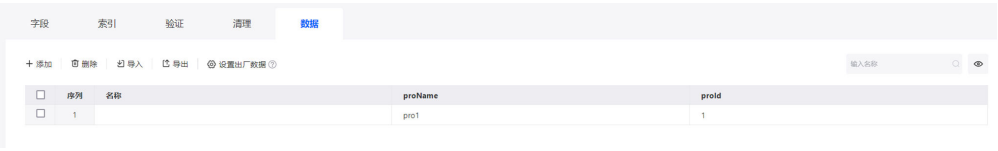


图 4-18 对象中有两条数据

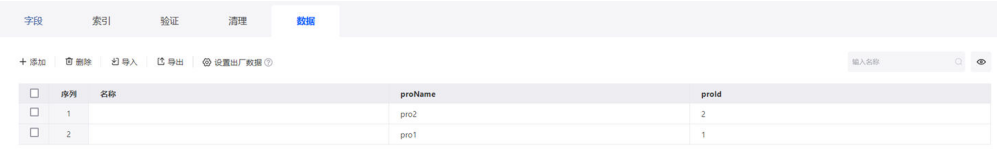
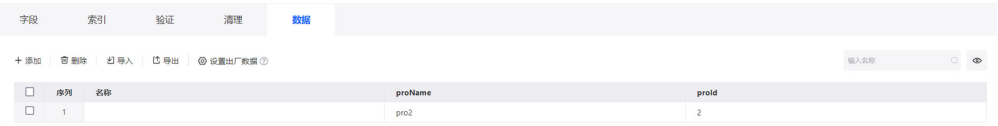


图 4-19 在前端页面删除 proId 为 1 的数据

图 4-20 对象中 proId 为 1 的数据同步删除



功能实现方法（增加功能实现）

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 4-21 创建一个空白应用

表 4-4 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建对象“product”，并为对象添加字段。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“product”，单击“确定”。

图 4-22 创建对象 product

表 4-5 新建 product 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	product

参数	说明	示例
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	product

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加ProName字段。

图 4-23 添加 ProName 字段

表 4-6 添加 proName 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	proName
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	proName
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

5. 在“字段”页签，再次单击“添加”按钮，添加proId字段。

图 4-24 添加 proId 字段

步骤3 创建一个脚本。

1. 在应用设计器中，选择“逻辑”，单击脚本后的“+”。
2. 新建一个空白的脚本，名称设置为“sqlAdd”，单击“添加”。

图 4-25 创建脚本 sqlAdd

3. 在脚本编辑器中，输入示例代码。

   本示例代码主要实现的功能为：在单击添加按钮时，获取前端输入的姓名、ID等数据，插入到对象中。如果插入失败，则记录失败信息。示例中的“命名空间__product__CST”为**步骤2**中创建的对象。

```
//本脚本用于创建工单
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库

//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    proId: string;
    @action.param({ type: 'String', required: true, label: 'String' })
    proName: string;
}
//定义出参结构，出参包含1个参数，workOrder的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
}
//使用数据对象命名空间__product__CST
@useObject(['命名空间__product__CST'])
@action.object({ type: "method" })
export class CreateWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public createWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let productData = new Object();
            productData['命名空间__proName__CST'] = input.proName;  //将入参赋值给productData变量，方便后面使用
            productData['命名空间__proId__CST'] = input.proId;
            let s = db.object('命名空间__product__CST');    //获取命名空间__product__CST这个Object的操作实例
            let id = s.insert(productData);
            if (id) {
```

```
            out.id = id;
        } else {
            error.name = "WOERROR";
            error.message = "Unable to create product!";
            throw error;
        }
    } catch (error) {
        console.error(error.name, error.message);
        context.setError(error.name, error.message);
    }
    return out;
  }
}
```

4. 单击  ，保存脚本，保存成功后单击  ，激活脚本。

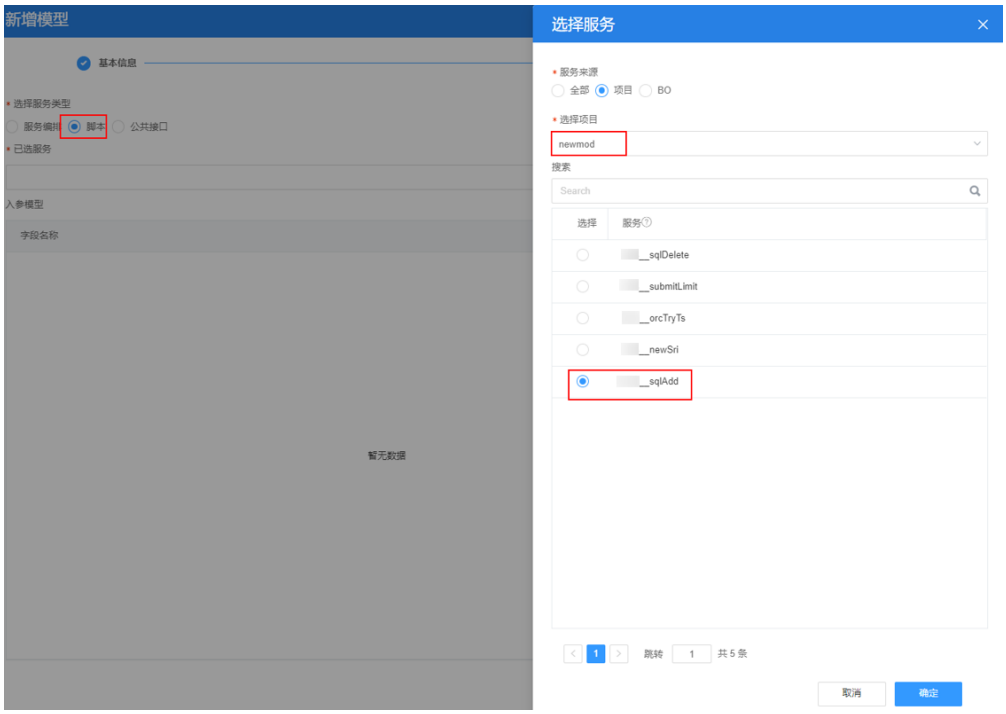
步骤4 新建对象模型。

1. 在应用设计器中，选择“界面”，单击页面后的“+”，新建一个标准页面。
2. 在标准页面底部，单击“模型视图”。
3. 单击“新增模型”，输入模型名称（如sqlAdd）、“来源”选择“服务”，单击“下一步”。

图 4-26 新建模型

4. 选择步骤3中创建的脚本，单击“确定”。

图 4-27 选择脚本

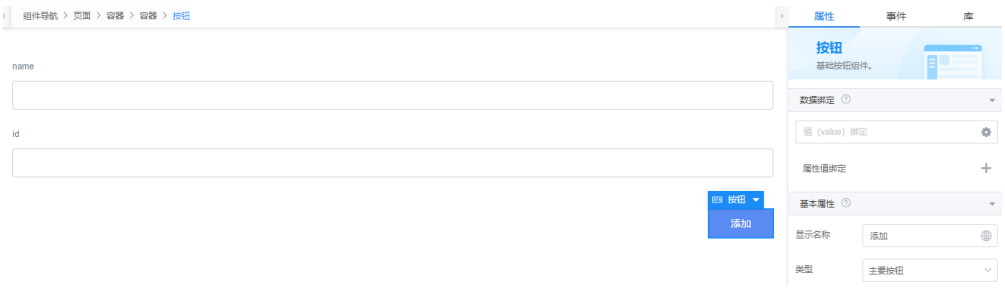


5. 单击“下一步”，再单击“确定”，完成模型的创建。

步骤5 返回设计视图页面，绑定关联模型。

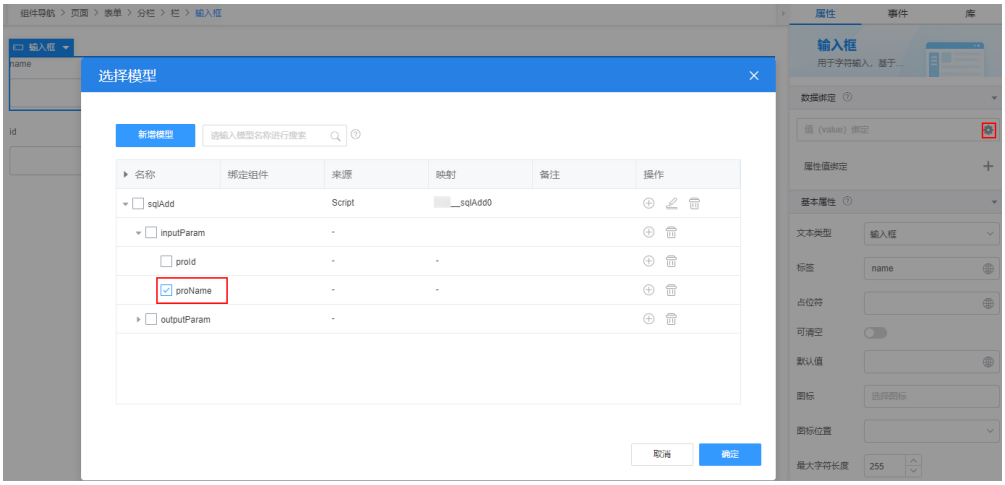
1. 在标准页面中，拖入2个输入框组件和1个按钮组件，将输入框标签修改为“name”和“id”，将按钮显示名称设置为“添加”。

图 4-28 页面最终设置效果



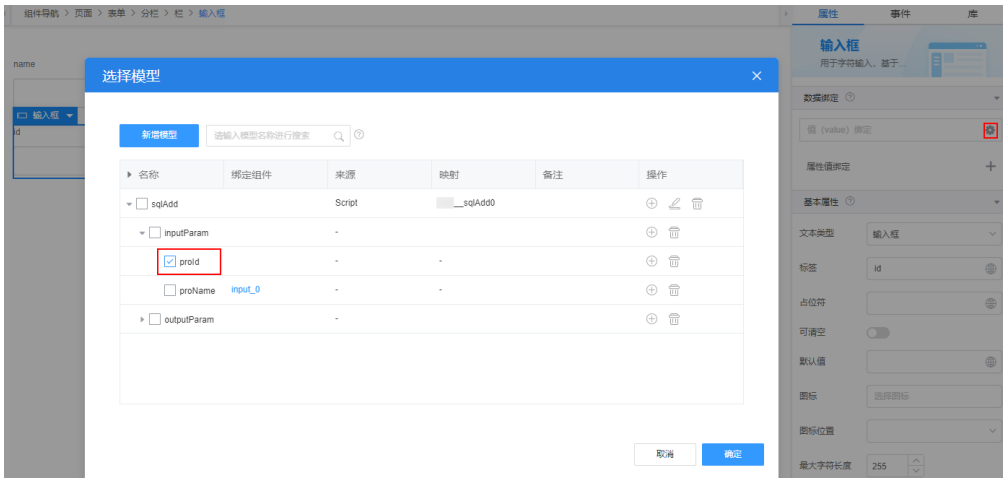
2. 选中name输入框，在“属性 > 数据绑定 > 值绑定”中，单击  。
3. 选中**步骤4**中创建的模型（proName），将输入框和脚本中的数据做绑定。

图 4-29 选择模型



4. 按照上述操作，为id输入框绑定**步骤4**中创建的模型（proId）。

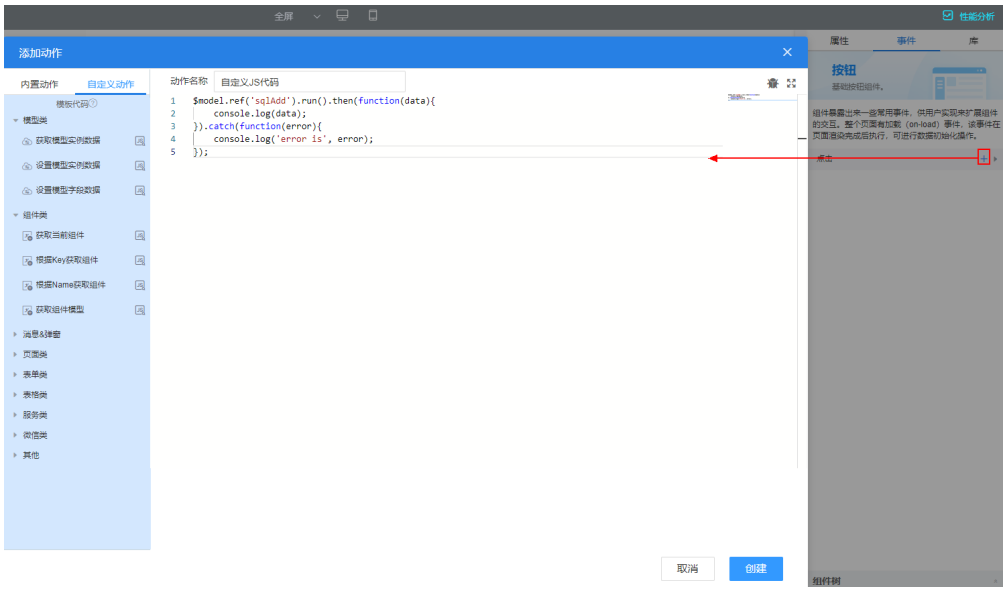
图 4-30 为 id 输入框绑定模型



步骤6 为添加按钮，添加事件。

1. 选中添加按钮组件，选择“事件”页签。
2. 单击“点击”后的“+”，进入添加动作页面。
3. 在自定义动作中，输入自定义代码，单击“创建”。

图 4-31 自定义动作



本示例中自定义的JS代码主要实现的功能为：单击提交按钮后，调用服务，服务调用脚本进行记录新增。

```
$model.ref('sqlAdd').run().then(function(data){
    console.log(data);
}).catch(function(error){
    console.log('error is', error);
});
```

步骤7 返回标准页面，单击  ，保存页面，保存成功后单击  ，预览效果。

----结束

功能实现方法（删除功能实现）

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 4-32 创建一个空白应用

表 4-7 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建一个删除脚本。

1. 在应用设计器中，选择“逻辑”，单击脚本后的“+”。
2. 新建一个空白的脚本（如sqlDelete），单击“添加”。

图 4-33 创建脚本 sqlDelete

3. 在脚本编辑器中，输入示例代码。

   本示例脚本主要实现的功能为：根据页面输入的ID参数，使用接口，根据ID按条件删除一条数据记录。如果报错，则记录错误信息。示例中的“命名空间__product__CST”为步骤2中创建的对象。

```
//本脚本用于删除工单
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
//定义入参结构
```

```
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    id: string;
}
//定义出参结构，出参包含1个参数，workOrder的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
}
//使用数据对象命名空间__product__CST
@useObject(['命名空间__product__CST'])
@action.object({ type: "method" })
export class DeleteWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public deleteWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let id = input.id;
            let s = db.object('命名空间__product__CST');    //获取命名空间__product__CST这个Object的操作
实例
            //查询条件
            let condition = {
                "conjunction": "AND",
                "conditions": [{
                    "field": "命名空间__proId__CST",
                    "operator": "eq",
                    "value": id
                }]
            };
            let isDeleted = s.deleteByCondition(condition);
            if (isDeleted) {
                out.id = id;
            } else {
                error.name = "WOERROR";
                error.message = "Failed to delete the work order!";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

4. 单击  ，保存脚本，保存成功后单击  ，激活脚本。

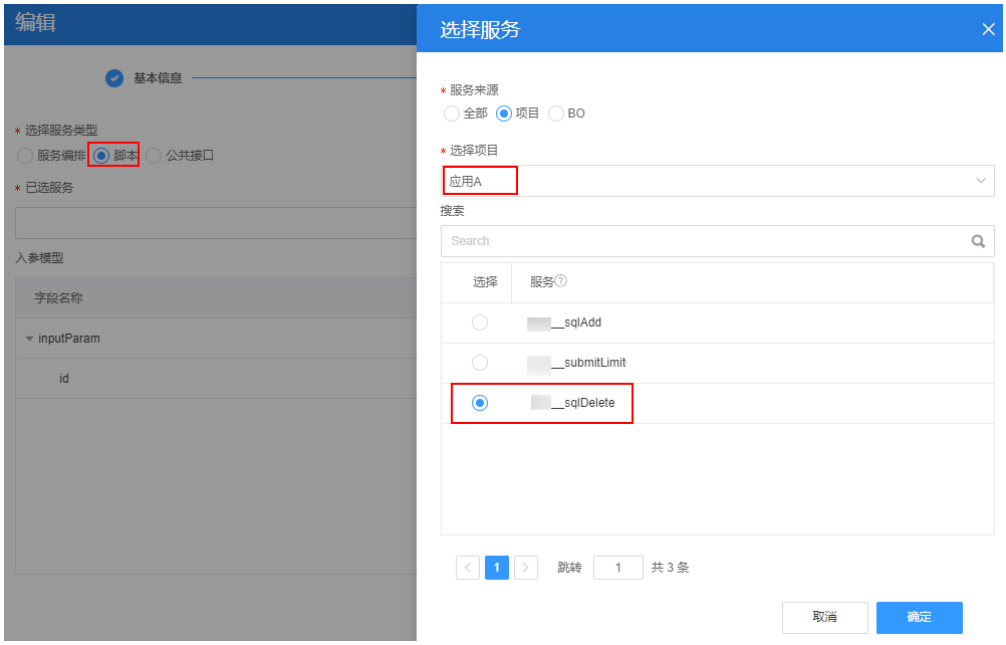
步骤3 新建对象模型。

1. 在应用设计器中，选择“界面”，单击页面后的“+”。
2. 输入页面的标签和名称，单击“添加”，新建一个标准页面。
3. 在标准页面底部，单击“模型视图”。
4. 单击“新增模型”，输入模型名称（如sqlDelete）、“来源”选择“服务”，单击“下一步”。

图 4-34 新建模型

5. “选择服务类型”设置为“脚本”，在弹出的页面选择**步骤2**中创建的脚本，单击“确定”

图 4-35 选择脚本

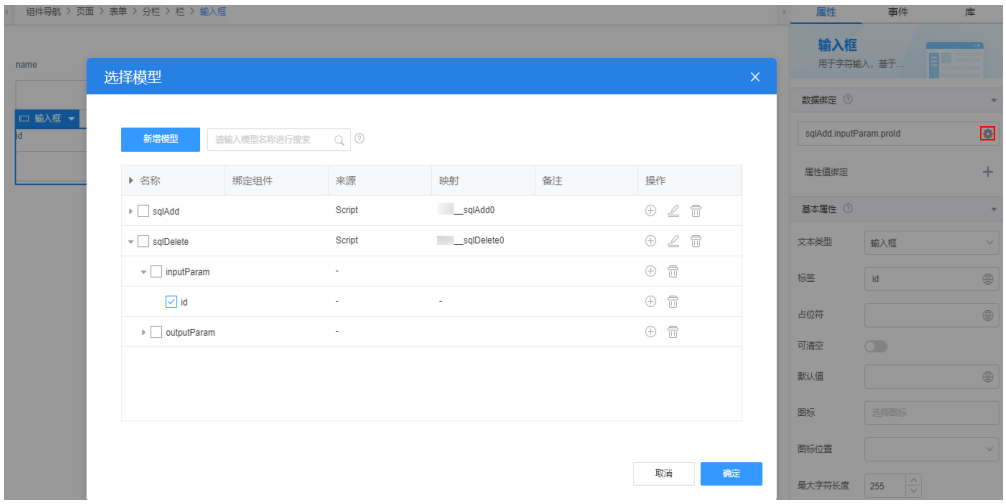


6. 单击“下一步”，再单击“确定”，完成模型的创建。

步骤4 返回设计视图页面，绑定关联模型。

1. 在标准页面中，拖入一个输入框组件和一个按钮组件，将输入框组件的“标签”设置为“id”，按钮组件的“显示名称”设置为“删除”。
2. 选中输入框组件，在“属性 > 数据绑定 > 值绑定”中，单击 。
3. 选择**步骤3**中创建的模型，输入框数据绑定选择sqlDelete模型中的入参“id”，单击“确定”。

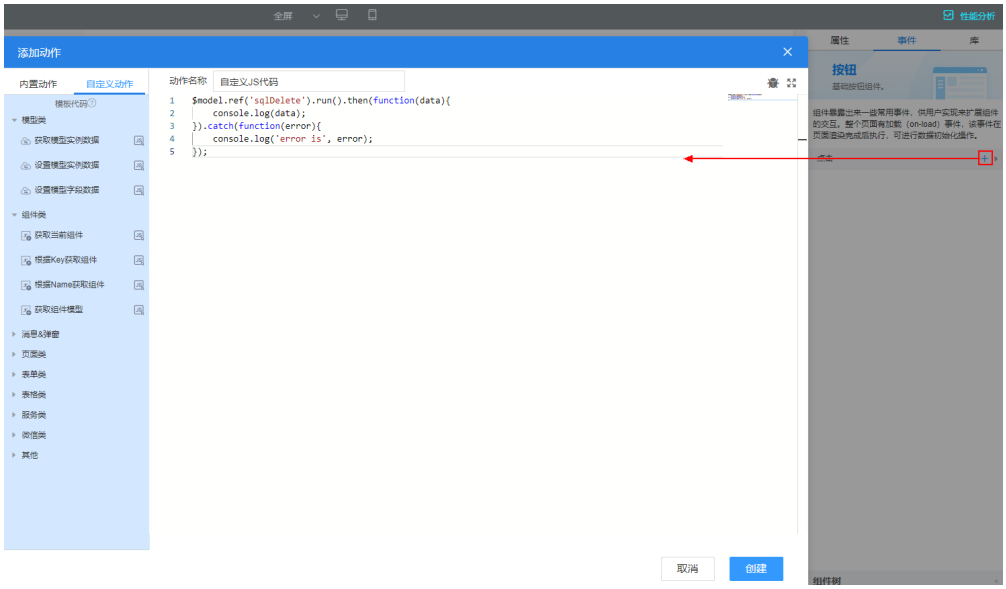
图 4-36 选择模型



步骤5 为删除按钮，添加事件。

1. 选中删除按钮组件，选择“事件”页签。
2. 单击“点击”后的“+”，进入添加动作页面。
3. 在自定义动作中，输入自定义代码，单击“创建”。

图 4-37 自定义动作



示例中自定义的JS代码主要实现的功能为：利用JS调用后端服务，后端服务调用脚本完成删除功能。

```
$model.ref('sqlDelete').run().then(function(data){
    console.log(data);
}).catch(function(error){
    console.log('error is', error);
});
```

步骤6 单击页面上方的，保存页面。

步骤7 保存成功后，单击页面上方的  ，预览效果。

----结束

5 模板专项

5.1 使用 AstroZero 文件模板生成合同文档

应用场景

AstroZero低代码平台提供了一个文件模板功能，基于该功能开发者可以根据不同客户设置不同类型的打印模板。例如，用户在商品订单管理系统中，基于商品、价格等信息，自动生成了商品订单列表。在处理完自己的订单后，可以使用模板功能将订单信息同步到合同模板中，生成合同文档，用于签订线下合同。同样，在财务领域，可以使用文档模板来生成定制化的发票和收据。对于需要发送正式邀请的场合，文档模板可以用来创建和打印专业的商务信函。

方案优势

用户可以在服务编排中，通过拖、拉、拽的方式调用文档模板节点、配置模板的输入输出参数，用于生成具体的文档。同时可以将该服务编排包装成开放接口供第三方使用，也可以在AstroZero的标准页面中直接调用供业务用户下载。

约束与限制

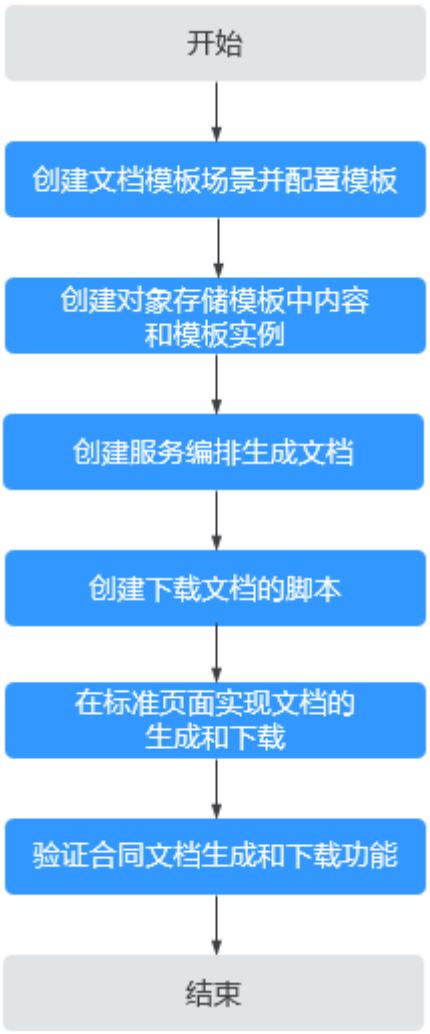
用户上传Word文档后（文档中包含使用${*参数*}定义的文本类型的变量），只能预览该文档模板或生成具体的文档，不能直接在界面上对其进行修改。如果需要，只能在本地更改Word模板样式后重新上传。

目前模板功能只支持生成docx类型的文档模板，并且只支持文本类的变量替换，无法动态替换二维码、图片等内容。

操作流程

在AstroZero中，通过文件模板生成具体合同文档的操作流程，如**图5-1**所示。

图 5-1 使用文件模板生成合同文档操作流程



步骤一：创建文档模板场景并配置模板

创建一个模板场景并在场景中添加一个合同模板。在模板场景的配置中，添加文档模板中需要替换的参数。模板场景是一个业务场景的集合，在模板场景中支持创建多个模板，模板之间共享数据结构。

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 5-2 创建一个空白应用

表 5-1 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建文件模板场景。

1. 在应用设计器的左侧导航栏中，选择“逻辑”，单击“更多 > 模板”。

图 5-3 单击模板

2. 单击“添加模板场景”，设置模板场景的标签和名称，单击“确定”。

图 5-4 创建模板场景

表 5-2 新建模板场景参数说明

参数	说明	示例
标签	新建模板场景的名称，创建后可修改。 取值范围：1~80个字符。	文档模板
名称	新建模板场景在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过64个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	docModel

步骤3 在“配置”页签，设置模板文件的存储位置，单击“保存”。

图 5-5 设置文件存储

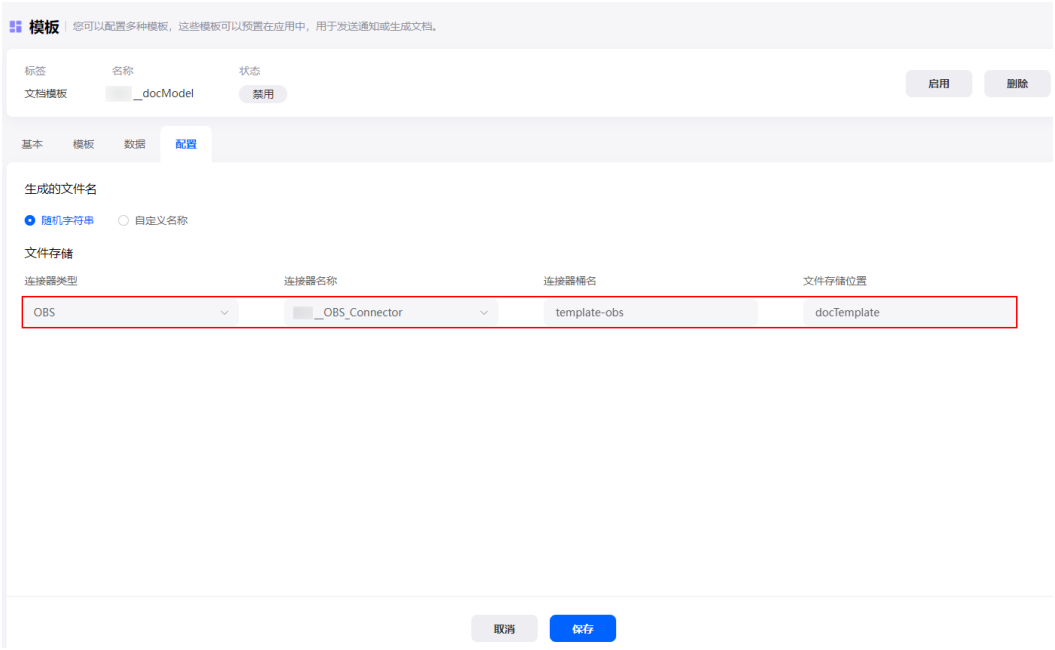


表 5-3 文件存储参数说明

参数	说明	示例
连接器类型	文件存储的连接器类型，当前仅支持“OBS”和“MINIO”两种。在AtroZero中，通过创建OBS、MINIO连接器，可将数据存储到OBS和MINIO中，详细介绍请参见**对接OBS云对象存储实例**、**对接MINIO存储AstroZero对象或资产**。	OBS
连接器名称	在AstroZero中创建OBS或MINIO连接器的名称，可在“集成 > 连接器 > 连接器实例 > 存储 > OBS/MINIO”中查看。	*命名空间*__OBS_Connector
连接器桶名	创建连接器时，配置的OBS或MINIO桶名称。如何查看OBS桶信息，请参见**查看桶信息**	template-obs
文件存储位置	指定文件在OBS或MINIO桶中的存储路径。	docTemplate

步骤4 在“模板”页签，单击“添加”，创建合同模板。

图 5-6 新建合同模板

表 5-4 新建模板参数说明

参数	说明	示例
标签	新建模板的名称，创建后可修改。 取值范围：1~80个字符。	合同档模板
名称	新建模板在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	contractModel

步骤5 在“数据”页签，为合同文档模板添加表5-5中参数。

此处添加的模板参数，对应文档模板中需要替换的内容。

图 5-7 为合同模板添加公司名称参数

表 5-5 待添加参数

名称	唯一标识	数据类型
公司名称	companyName	文本
合同金额	amount	数字
订单数目	orderNum	数字
合同签订人	person	文本
合同日期	date	日期
合同名称	contractName	文本
乙方公司名称	otherCompanyName	文本

步骤6 上传文档模板。

1. 在“模板”页签，单击**步骤4**中创建模板后的，进入合同文档模板页面。

图 5-8 进入模板编辑页面

2. 单击“点击上传”，选择本地待上传的word文档，查看上传后效果。

说明

上传操作执行成功后，在**步骤3**中配置的OBS桶中可查看到已上传的word文档，如**图5-11**所示。

图 5-9 上传 word 文档

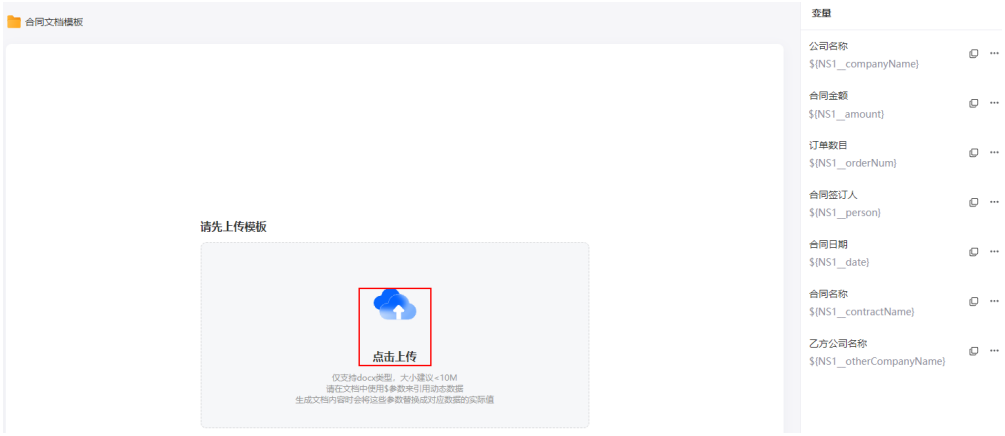


图 5-10 查看文档上传后效果

图 5-11 在 OBS 桶中查看文档是否上传

本实践中待上传的word文档内容如下，实际使用时请根据自身的业务需求进行上传。

${*命名空间*__contractName}

第一条合同目的

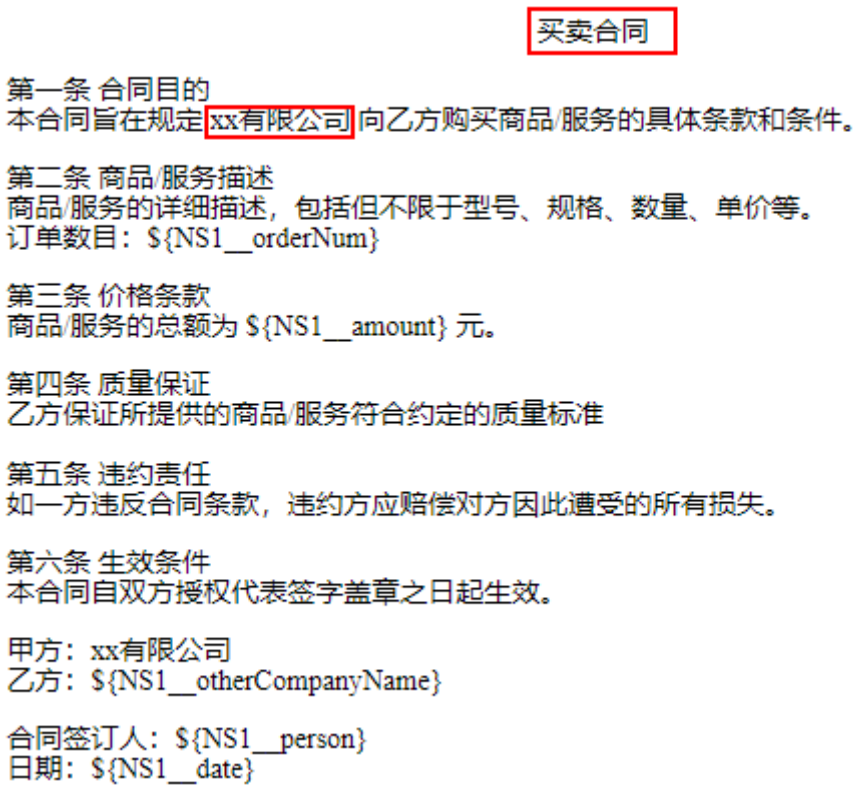
```
本合同旨在规定${命名空间__companyName} 向乙方购买商品/服务的具体条款和条件。

第二条商品/服务描述
商品/服务的详细描述，包括但不限于型号、规格、数量、单价等。
订单数目：${命名空间__orderNum}

第三条价格条款
商品/服务的总额为 ${命名空间__amount} 元。

第四条质量保证
乙方保证所提供的商品/服务符合约定的质量标准

第五条违约责任
如一方违反合同条款，违约方应赔偿对方因此遭受的所有损失。

第六条生效条件
本合同自双方授权代表签字盖章之日起生效。

甲方：${命名空间__companyName}
乙方：${命名空间__otherCompanyName}

合同签订人：${命名空间__person}
日期：${命名空间__date}
```

3. 单击页面右上方的“预览”，输入参数，预览效果。

   例如，在“输入参数”中输入如下模板参数，查看合同中对应参数是否被替换。

```
{
   "命名空间__contractName": "买卖合同",
   "命名空间__companyName": "xx有限公司"
}
```

图 5-12 合同中参数已被替换



4. 预览符合预期后，单击页面右上方的“启用”，启用文档模板。

步骤7 返回文档模板场景中，单击“启用”，启用模板场景。

图 5-13 启用模板场景

----结束

步骤二：创建对象存储模板中内容和模板实例

创建一个对象并在对象中添加字段，用于存储文档模板中的内容和后续生成的模板实例。

步骤1 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。

步骤2 设置对象的名称和唯一标识，单击“确定”。

图 5-14 创建对象 docObject

表 5-6 新建 docObject 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	文档模板对象

参数	说明	示例
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	docObject

步骤3 在已创建的对象中，单击，进入对象详情页面。

图 5-15 选择编辑按钮

步骤4 在“字段”页签，单击“添加”，为对象添加companyName字段。

图 5-16 添加 companyName 字段

表 5-7 添加 companyName 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	公司名称
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 • 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	companyName
字段类型	单击 ••• ，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本
数据长度	允许输入字段的长度。	64

步骤5 按照上述操作，为对象继续添加表5-8中字段。

图 5-17 查看对象中字段

表 5-8 待添加对象字段

名称	唯一标识	数据类型
公司名称（已添加）	companyName	文本
合同金额	amount	数字
订单数目	orderNum	数字
合同签订人	person	文本
合同日期	date	日期
合同名称	contractName	文本
乙方公司名称	otherCompanyName	文本
合同模板实例	modedoc	文本（数据长度设置为255）

----结束

步骤三：创建服务编排生成文档

创建一个服务编排，添加“生成文档”和“记录创建”两个图元，用于根据合同中参数创建具体的文档。

步骤1 在应用设计器的左侧导航栏中，选择“逻辑”，单击编排后的“+”。

步骤2 设置服务编排的标签和名称，单击“添加”。

图 5-18 新建服务编排

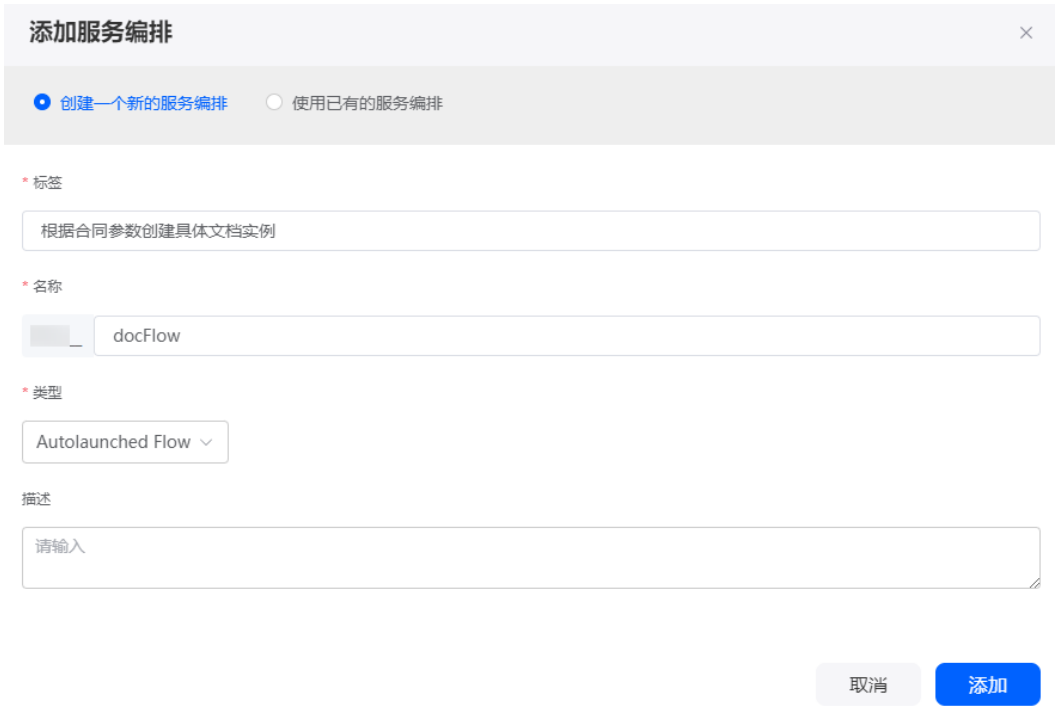


表 5-9 新建服务编排参数说明

参数	说明	示例
标签	服务编排的标签名，用于在界面展示，创建后可修改。 取值范围：1~64个字符。	根据合同模板创建具体文档实例
名称	服务编排在系统中的唯一标识，创建后不支持修改。命名要求如下： • 长度不能超过64个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 名称必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不能以下划线结尾。	docFlow

步骤3 创建全局上下文变量。

1. 在服务编排设计页面，选中开始节点，单击 。
2. 在全局上下文中，单击变量后的，新建变量variable0。
3. 单击variable0变量后的，选择“设置”。
4. 将变量的“名称”设置为“companyName”，单击“保存”。

图 5-19 新建 companyName 变量

5. 按照上述操作，创建表5-10中的变量。

图 5-20 查看已创建的变量

表 5-10 需要创建的全局上下文变量

名称	数据类型
companyName（已创建）	文本
amount	数字
orderNum	数字
person	文本
date	日期
contractName	文本
otherCompanyName	文本
modedoc	文本

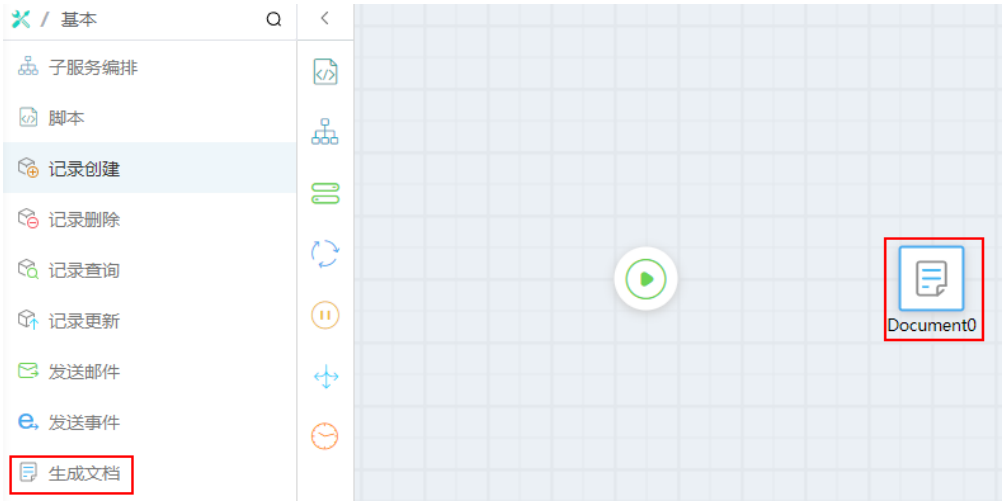
6. 选中开始节点，设置节点的输入参数和输出参数。

图 5-21 设置入参和出参

步骤4 添加生成文档节点。

1. 在“基本”图元中，拖拽“生成文档”图元到开始图元后。

图 5-22 拖拽生成文档图元到画布中



2. 选中生成文档图元，单击  ，设置生成文档。

图 5-23 设置生成文档图元

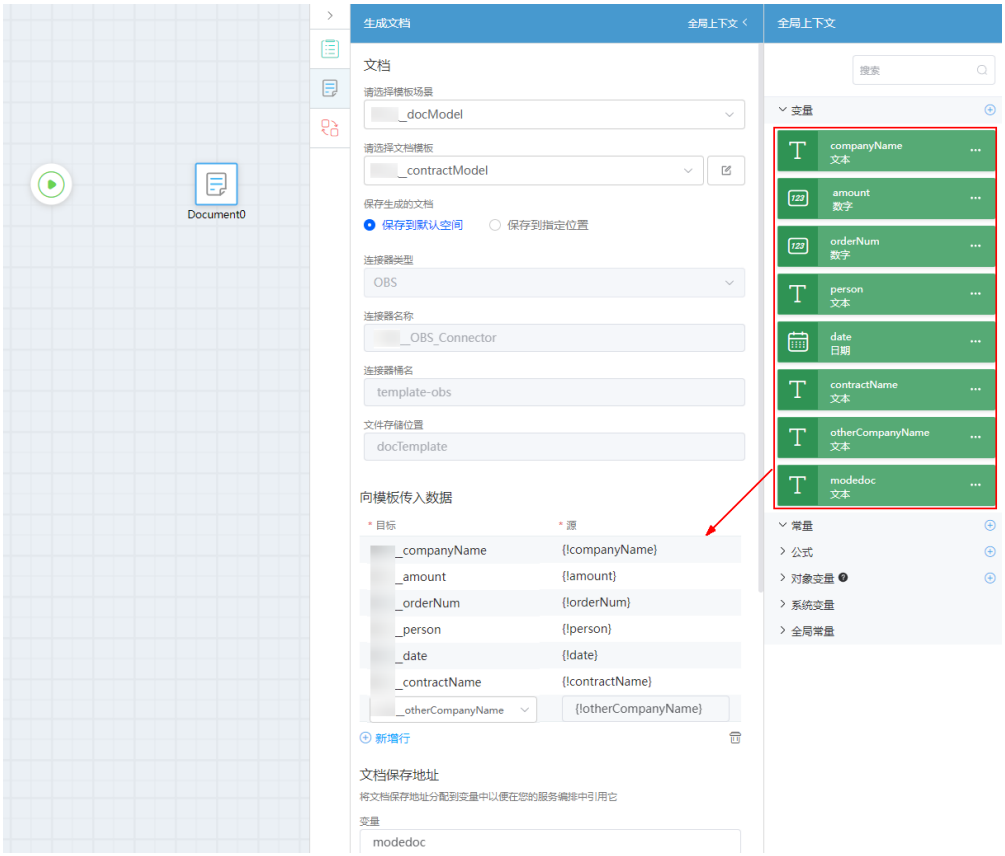


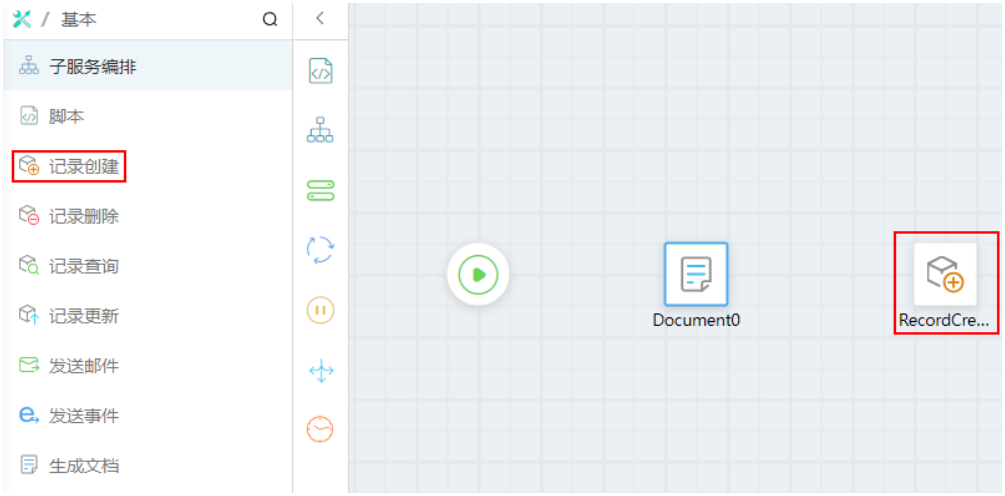
表 5-11 生成文档图元设置说明

参数	说明	示例
请选择模板场景	选择生成文档图元关联的模板场景，即**步骤2**中创建的。	*命名空间*__docModel
请选择文档模板	选择模板场景中创建的文档模板，即**步骤4**中创建的。	*命名空间*__docModel
连接器类型	根据选择的文档模板自动进行关联。	OBS
连接器名称	根据选择的文档模板自动进行关联。	*命名空间*__OBS_Connector
连接器桶名	根据选择的文档模板自动进行关联。	template-obs
向模板传入数据	向模板中传入数据，将入参变量一次赋值给对应的模板参数。	**步骤3**中创建的变量
文档保存地址	将出参modelDoc变量放入文档保存地址，用于存放生成的文档名。	modedoc

步骤5 添加创建记录节点。

1. 在“基本”图元中，拖拽“记录创建”图元到“生成文档”图元后。

图 5-24 添加记录创建图元



2. 选中记录创建图元，单击  ，将入参、出参变量都存入**步骤二：创建对象存储模板中内容和模板实例**中创建的对象中。

图 5-25 记录创建设置

步骤6 连接图元指定逻辑关系，即从开始图元连向生成文档图元，从生成文档图元连向记录创建图元。

图 5-26 指定图元逻辑关系

步骤7 单击页面上方的，保存服务编排。

步骤8 单击，进入服务编排调试页面。

在输入参数中，输入如下内容，并单击“运行”。

```
{
   "companyName": "A公司",
   "amount": "10",
   "orderNum": "2",
   "person": "张三",
   "date": "2024-11-05",
   "contractName": "新采购合同",
   "otherCompanyName": "B公司"
}
```

执行成功后，提示如下信息。登录OBS服务控制台，在**步骤3**中配置的路径下会生成**图5-27**中的合同文档。将合同文档下载到本地，查看合同中内容，如**图5-29**所示，可发现合同中的内容已被替换。

图 5-27 服务编排执行成功

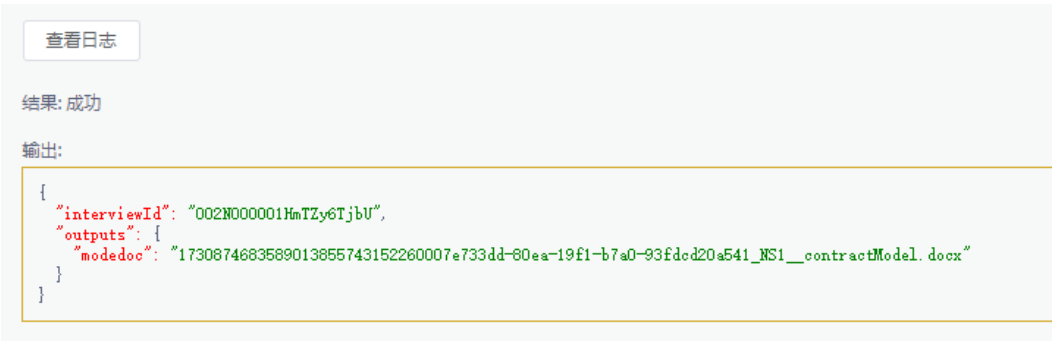


图 5-28 查看新生成的合同文档

图 5-29 查看合同内容

新采购合同

第一条 合同目的
本合同旨在规定 A公司 向乙方购买商品/服务的具体条款和条件。

第二条 商品/服务描述
商品/服务的详细描述，包括但不限于型号、规格、数量、单价等。
订单数目：2

第三条 价格条款
商品/服务的总额为 10 元。

第四条 质量保证
乙方保证所提供的商品/服务符合约定的质量标准

第五条 违约责任
如一方违反合同条款，违约方应赔偿对方因此遭受的所有损失。

第六条 生效条件
本合同自双方授权代表签字盖章之日起生效。

甲方：A公司
乙方：B公司

合同签订人：张三
日期：2024-11-05

步骤9 单击页面上方的，启用服务编排。

----结束

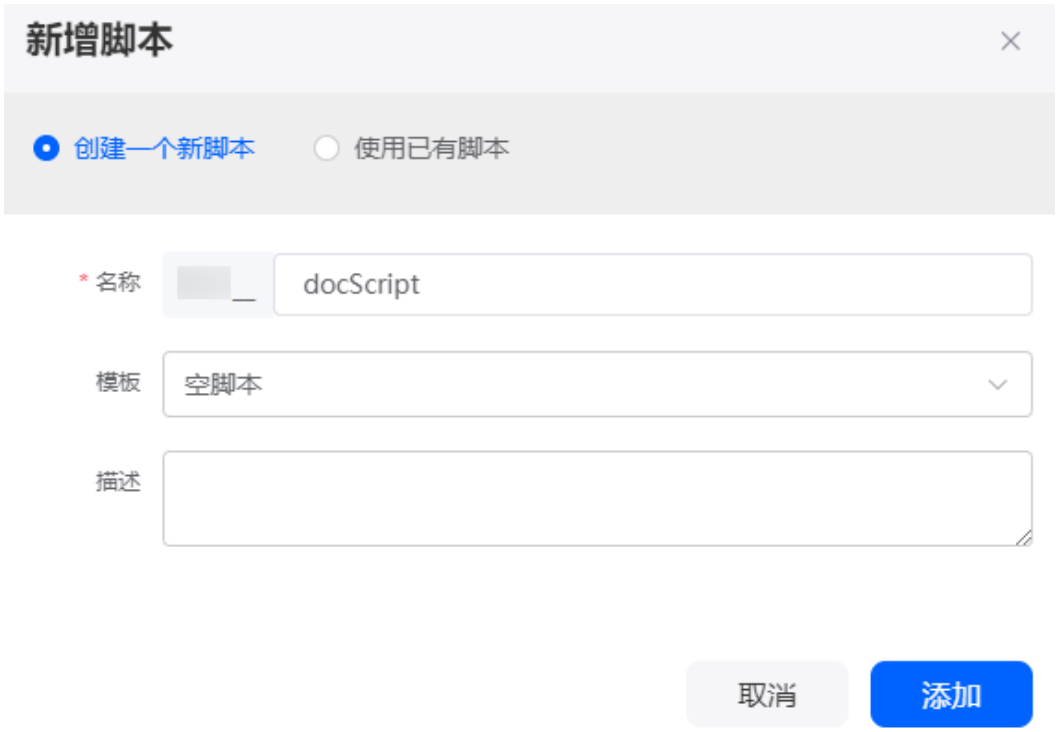
步骤四：创建下载文档的脚本

创建一个脚本，用于根据文档名下载OBS桶中生成的合同文档。

步骤1 在应用设计器的左侧导航栏中，选择“逻辑”，单击脚本后的“+”。

步骤2 新建一个空白的脚本，名称设置为“docScript”，单击“添加”。

图 5-30 创建脚本 docScript



步骤3 在脚本编辑器中，输入示例代码。

本示例代码主要用于下载文档，示例中的“命名空间__OBS_Connector”为**步骤3**中配置的连接器名称，“docTemplate”为**步骤3**中配置的文件存储位置。

```
import * as context from 'context';//导入上下文相关的标准库
import * as objectstorage from 'objectstorage';


//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    docName: string;
}
//定义出参结构
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'Any' })
    buf: any;
}

@action.object({ type: "method" })
export class CreateWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public createWorkOrder(input: ActionInput): ActionOutput {
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            // OBS桶路径，和模板配置中的一致
            let path = "docTemplate/";
            // 调用连接器下载，NS1__OBS_Connector为调用连接器名称
            let obsCli = objectstorage.newClient(objectstorage.StoreType.OBS, "命名空间__OBS_Connector");
            let data = obsCli.getObject(path + input.docName);
            out.buf = data;
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
```

```
    }
    return out;
  }
}
```

步骤4 单击，保存脚本。

步骤5 运行测试脚本。

1. 单击编辑器上方的，执行脚本。
2. 设置输入参数，单击测试窗口右上角的，查看返回消息。

```
{
  "docName": "173087468358901385574315226O007e733dd-80ea-19f1-b7a0-93fdcd20a541_命名空间__contractModel.docx"
}
```

其中，“173087468358901385574315226O007e733dd-80ea-19f1-b7a0-93fdcd20a541_命名空间__contractModel.docx”为步骤三：创建服务编排生成文档中生成的文档，即通过脚本根据文档名下载OBS桶中的实际合同文档。

图 5-31 查看输出参数

步骤6 脚本测试完成后，单击页面上方的，启用该脚本。

----结束

步骤 5：通过标准页面调用服务编排和脚本实现文档的生成和下载

设计一个标准页面，通过调用服务编排和脚本，在前端页面实现合同文档的生成和下载等功能。

步骤1 新建一个标准页面。

1. 在应用设计器中，选择“界面”，单击页面后的“+”，新建一个标准页面。
2. 设置标准页面的标签和名称，单击“添加”。

图 5-32 添加标准页面

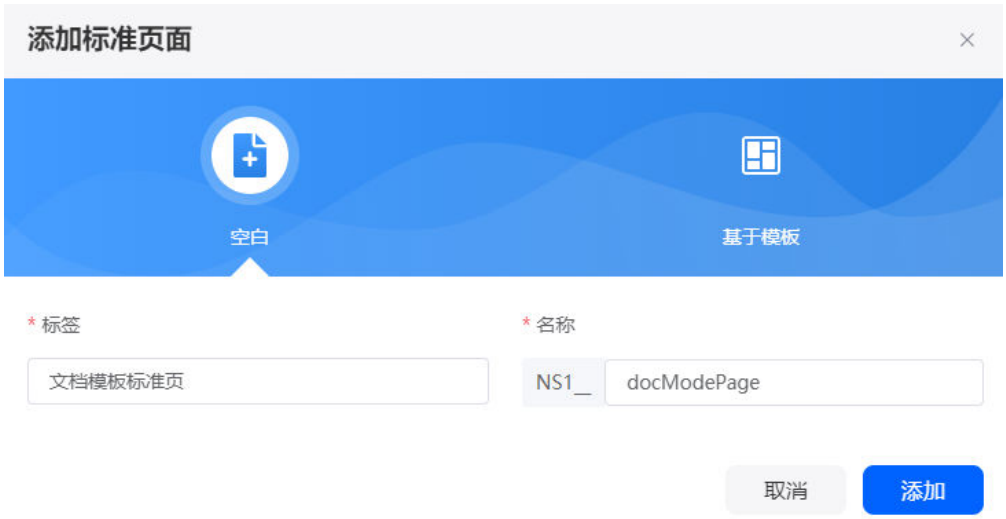


表 5-12 新建标准页面参数说明

参数	说明	示例
标签	输入标准页面的标签名，用于在页面显示，创建后可修改。 取值范围：1~64个字符。	文档模板标准页
名称	输入标准页面的名称，名称是标准页面在系统中的唯一标识，创建后不可修改。命名要求如下： – 长度不能超过64个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不能以下划线结尾。	docModePge

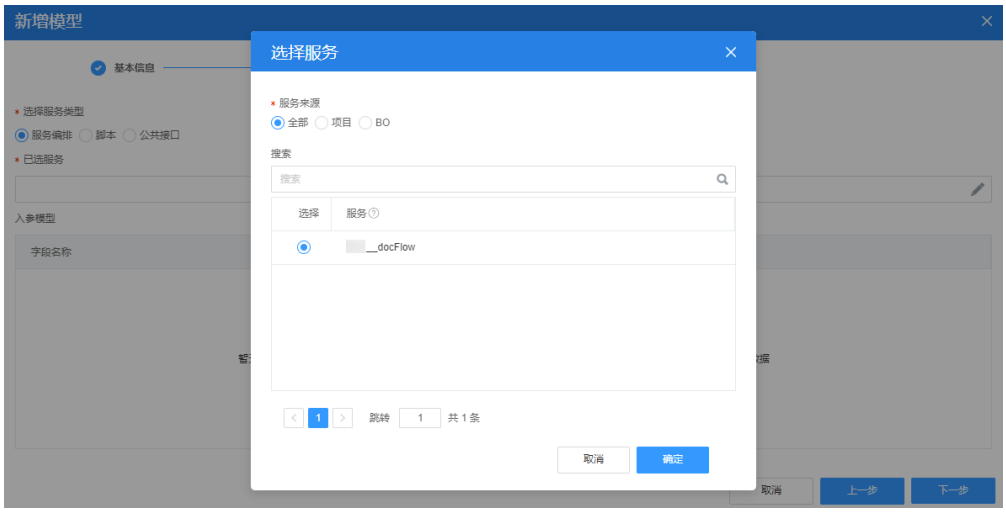
步骤2 新增对象模型。

1. 在标准页面底部，单击“模型视图”，将页面从设计视图切换到模型视图。
2. 单击“新增模型”，输入模型名称（flowDemo）、“来源”选择“服务”，单击“下一步”。

图 5-33 新增服务编排对象模型

3. 选择**步骤三：创建服务编排生成文档**中创建的服务编排，单击“下一步”，再单击“确定”，完成模型的创建。

图 5-34 选择目标服务编排



4. 再次单击“新增模型”，输入模型名称（scriptDemo）、“来源”选择“服务”，单击“下一步”。

图 5-35 新增脚本对象模型

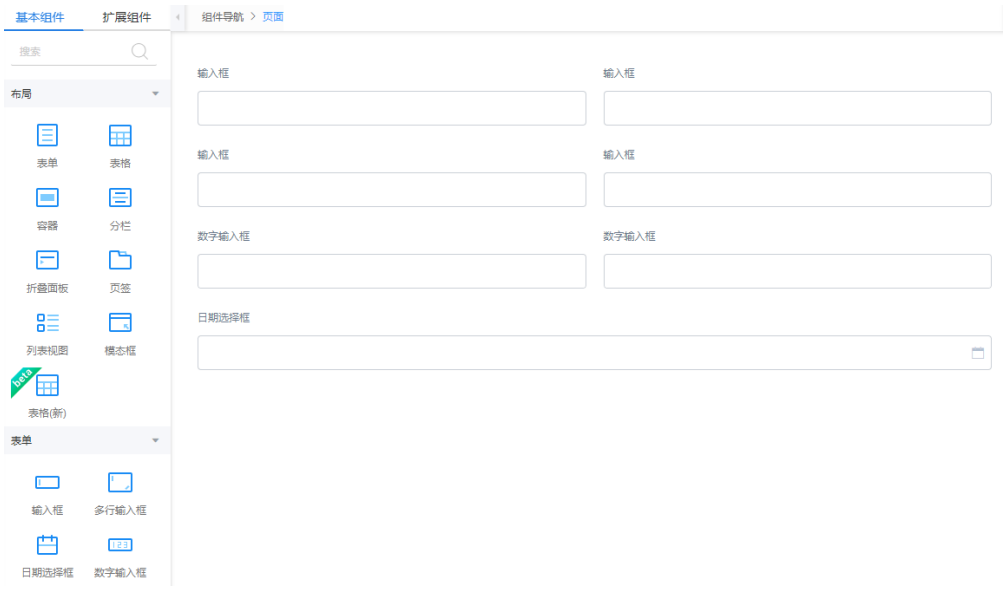
5. 选择**步骤四：创建下载文档的脚本**中创建的脚本，单击“下一步”，再单击“确定”，完成模型的创建。

图 5-36 选择目标脚本

步骤3 设计一个标准页面，用于生成和下载合同文档。

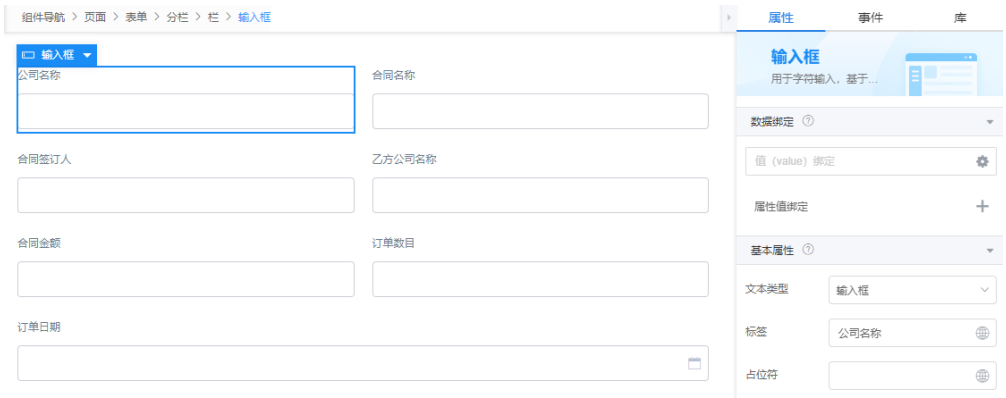
1. 在标准页面底部，单击“设计视图”，从模型视图切换到设计视图。
2. 从基本组件中，拖拽输入框（4个）、数字输入框（2个）和日期选择器到标准页面的画布中，布局如**图5-37**。

图 5-37 拖拽组件到画布



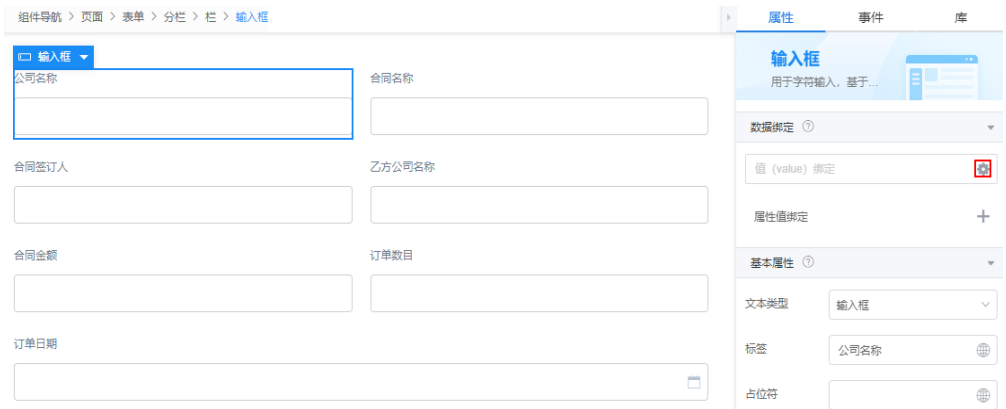
3. 选中组件，修改组件的标签，修改后效果如图5-38所示。

图 5-38 修改组件标签



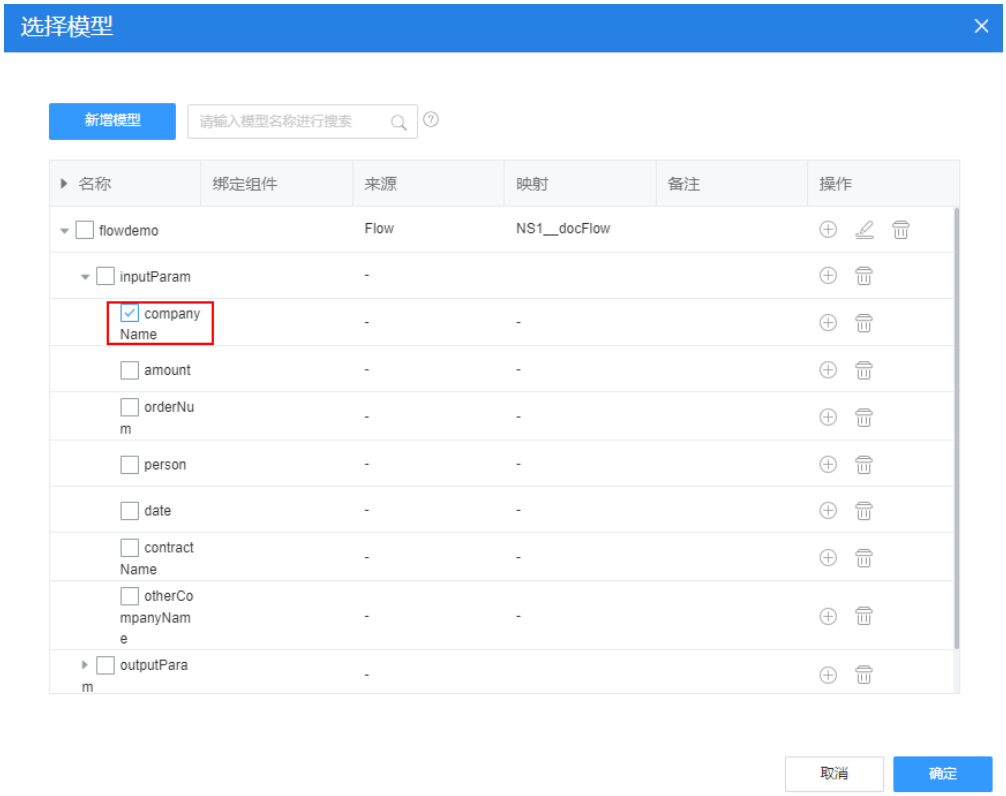
4. 选中公司名称输入框，在“属性 > 数据绑定”中，单击“值（value）绑定”后的 。

图 5-39 选择值绑定



5. 选择步骤2.2模型中的“companyName”，单击“确定”。

图 5-40 绑定 companyName

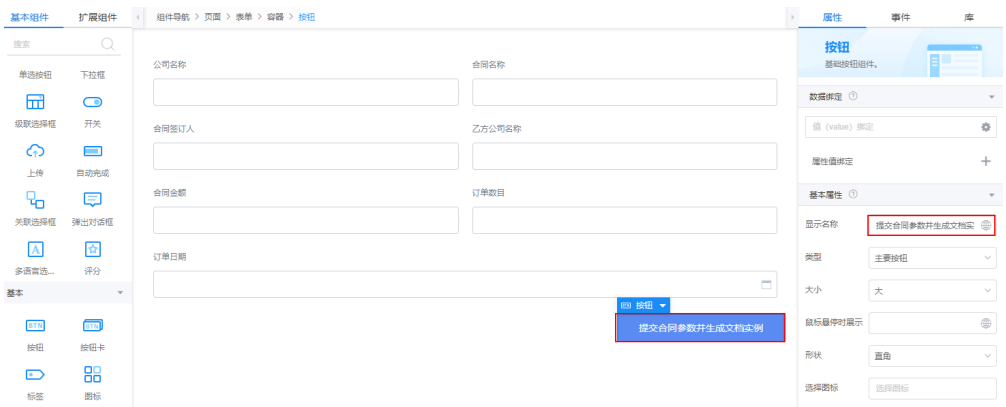


6. 按照上述操作，分别为其他组件绑定模型。
“合同名称”绑定“flowdemo.inputParam.contractName”，“合同签订人”绑定“flowdemo.inputParam.person”，“乙方公司名称”绑定“flowdemo.inputParam.otherCompanyName”，“合同金额”绑定“flowdemo.inputParam.amount”，“订单数目”绑定“flowdemo.inputParam.orderNum”，“订单日期”绑定“flowdemo.inputParam.date”。

步骤4 为标准页面添加一个按钮组件。

1. 从“基本组件 > 基本”中，拖拽一个按钮组件到订单日期组件下方，并将按钮的“显示名称”修改为“提交合同参数并生成文档实例”。

图 5-41 添加按钮组件



2. 选中按钮组件，在“事件”页签中，单击“点击”后的 ，进入添加动作页面。
3. 在自定义动作中，输入如下示例代码，单击“创建”。

   本示例代码实现的功能：根据服务编排模型中设置的合同参数，生成对应的合同文档。其中，“flowDemo”为步骤2.2中创建的服务编排模型。

```
$model.ref('flowDemo').run().then(function(data){
    console.log(data);
    context.$message.success('Submitted successfully.');
}).catch(function(error){
    context.$message.error('Submission failed:' + error.resMsg);
});
```

步骤5 为标准页面添加一个表格组件。

1. 从“基本组件 > 布局”中，拖拽一个表格组件到按钮组件下方。

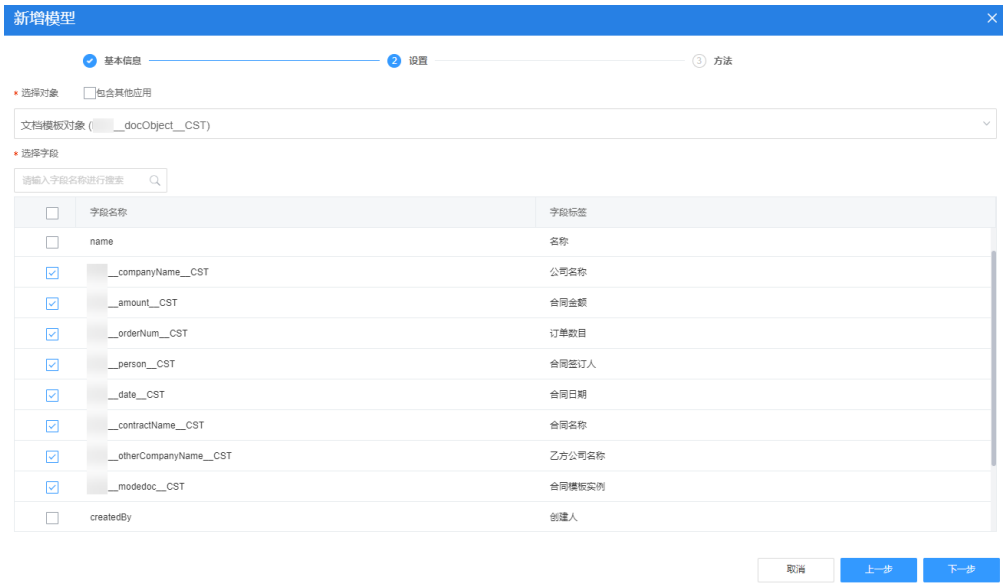
图 5-42 添加表格组件

2. 选中表格组件，在“属性 > 数据绑定”中，单击“值（value）绑定”后的 。
3. 单击“新建模型”，输入模型名称（obj）、“来源”选择“对象”，单击“下一步”。

图 5-43 设置模型基本信息

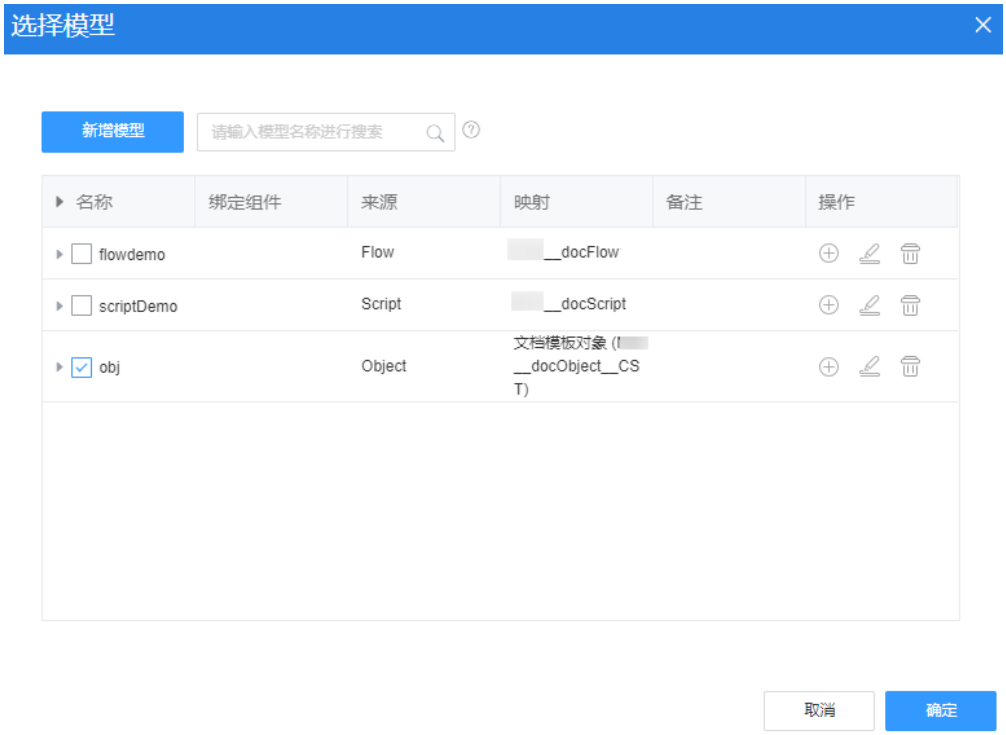
4. 选择**步骤二：创建对象存储模板中内容和模板实例**中创建的对象和字段，单击“下一步”，再单击“确定”。

图 5-44 模型设置



5. 在选择模型中，选中创建的模型，将对象模型绑定到表格上。

图 5-45 为表格绑定模型

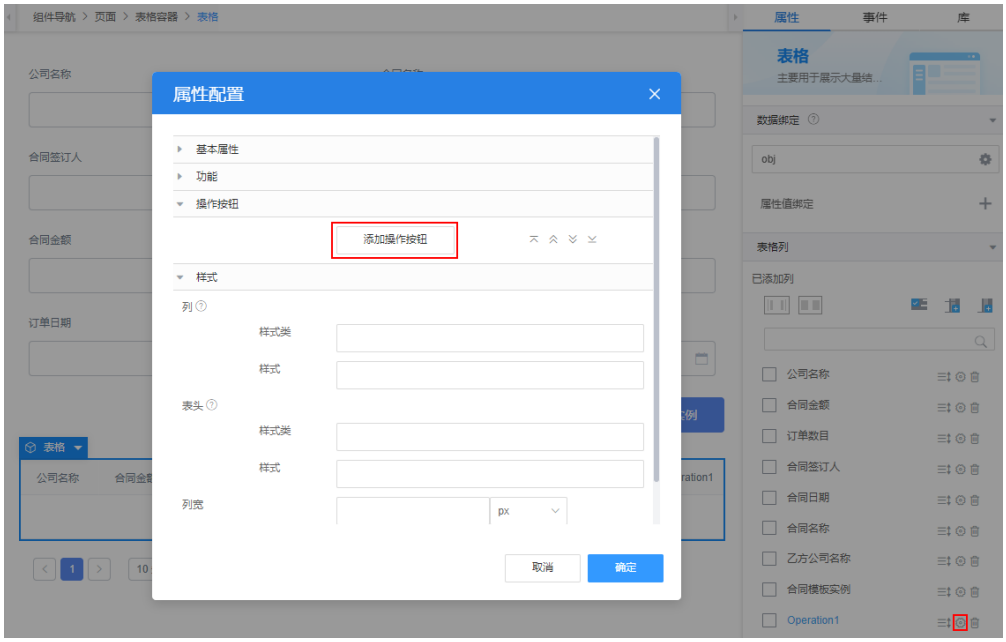


6. 选中表格组件，为表格添加一个操作列。

图 5-46 为表格添加操作列

7. 单击新建操作列后的，在操作按钮中，单击“添加操作按钮”。

图 5-47 添加操作按钮



8. 单击，单击“动作列表”后的，为按钮添加事件。

图 5-48 为按钮添加动作

9. 在自定义动作中，输入如下示例代码，单击“创建”。

   本示例代码主要实现的功能：根据合同模板实例的字段名，从OBS中下载服务编排生成的文档。其中，*“命名空间__modedoc__CST”*为合同模板实例的字段名，“scriptDemo”为**步骤2.4**中创建的脚本对象模型。

```
// base64解码函数
function base64DecodeToBinary(base64String) {
  // 解码Base64字符串
  const binaryString = atob(base64String);
  // 将解码后的字符串转换为二进制数据
  const len = binaryString.length;
  const bytes = new Uint8Array(len);
  for (let i = 0; i < len; i++) {
    bytes[i] = binaryString.charCodeAt(i);
  }
  // 将Uint8Array转换为Blob
  const blob = new Blob([bytes]);
  return blob;
}
// 获取表格中的行数据，命名空间__modedoc__CST为表格中合同模板实例的字段名
var rowData = context.$component.current.$attrs.row;
const docName = rowData.命名空间__modedoc__CST;
// 设置脚本入参
let originData = $model.ref('scriptDemo').getData();
originData.inputParam.docName = docName;
$model.ref('scriptDemo').setData(originData);
//运行脚本,从OBS中下载服务编排生成的文档
$model.ref('scriptDemo').run().then(function(data){
```

```
    const decodedString = base64DecodeToBinary(data.buf);
    const url = URL.createObjectURL(decodedString);
    let link = document.createElement('a');
    link.href = url;
    // 设置下载的文档名称，命名空间__modedoc__CST为表格中合同模板实例的字段名
    link.setAttribute('download', rowData.命名空间__modedoc__CST);
    document.body.appendChild(link);
    link.click();
    URL.revokeObjectURL(link.href);
    document.body.removeChild(link);
    context.$message.success('DownLoad successfully.');
}).catch(function(error){
    console.log('error is', error);
    context.$message.error('DownLoad failed:' + error.resMsg);
});
```

10. 将操作按钮的“标签”设置为“下载文档”，同时将基本属性中的“列标题”设置为“下载”。

图 5-49 更改操作列标题和操作的标签

11. 返回标准页面，在属“性 > 表格区块”中，单击工具栏后的“添加”，为表格添加一个工具栏。

图 5-50 为表格添加工具栏

步骤6 标准页面设计完成后，单击页面上方的，保存标准页面。

----结束

步骤六：验证合同文档生成和下载功能

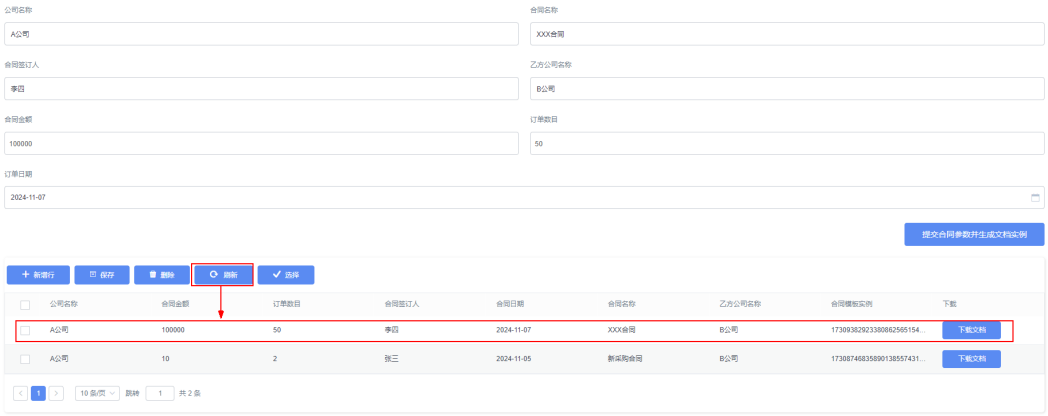
步骤1 在标准页面上方，单击，进入预览页面。

步骤2 输入合同内容，单击“提交合同参数并生成文档实例”。

图 5-51 输入合同内容

步骤3 提示“Submitted successfully”后，单击“刷新”，即可查看到提交的合同数据。

图 5-52 查看提交的合同数据



步骤4 单击数据后的“下载文档”，将合同文档下载到本地并查看文档内容。

图 5-53 合同文档被下载到本地

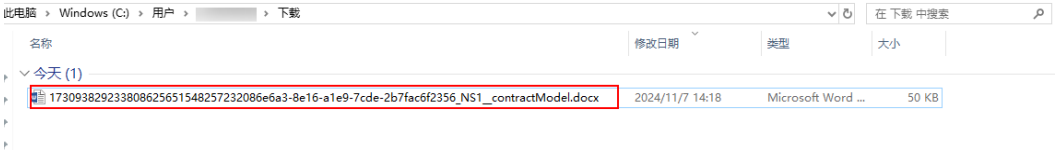


图 5-54 查看下载到本地合同的内容

XXX合同

第一条 合同目的
本合同旨在规定 A公司 向乙方购买商品/服务的具体条款和条件。

第二条 商品/服务描述
商品/服务的详细描述，包括但不限于型号、规格、数量、单价等。
订单数目：50

第三条 价格条款
商品/服务的总额为 100000 元。

第四条 质量保证
乙方保证所提供的商品/服务符合约定的质量标准

第五条 违约责任
如一方违反合同条款，违约方应赔偿对方因此遭受的所有损失。

第六条 生效条件
本合同自双方授权代表签字盖章之日起生效。

甲方：A公司
乙方：B公司

合同签订人：李四
日期：2024-11-07

----结束

6 工作流专项

6.1 通过 AstroZero 流程模板创建出差审批电子流

方案概述

AstroZero低代码平台基于业界BPMN 2.0标准，实现了自己的业务流程管理系统，即工作流。AstroZero中的工作流是一套图形化的流程编排引擎，着重于构建带有用户交互行为的业务流程，例如审批流、工单派发流程等。本实践通过创建一个出差审批应用，帮助您快速熟悉AstroZero中的工作流。

员工出差是企业运营中最常见的业务场景，涉及到一系列的管理流程，例如出差申请、审批、行程安排、费用报销等。本实践以一个简单的员工出差场景，即员工在出差前需要提交一个出差申请审批的电子流程，员工提交出差申请后，主管处理审批或拒绝提交人申请，向您介绍如何使用AstroZero中的工作流。本示例中的出差审批应用主要包括如下功能：

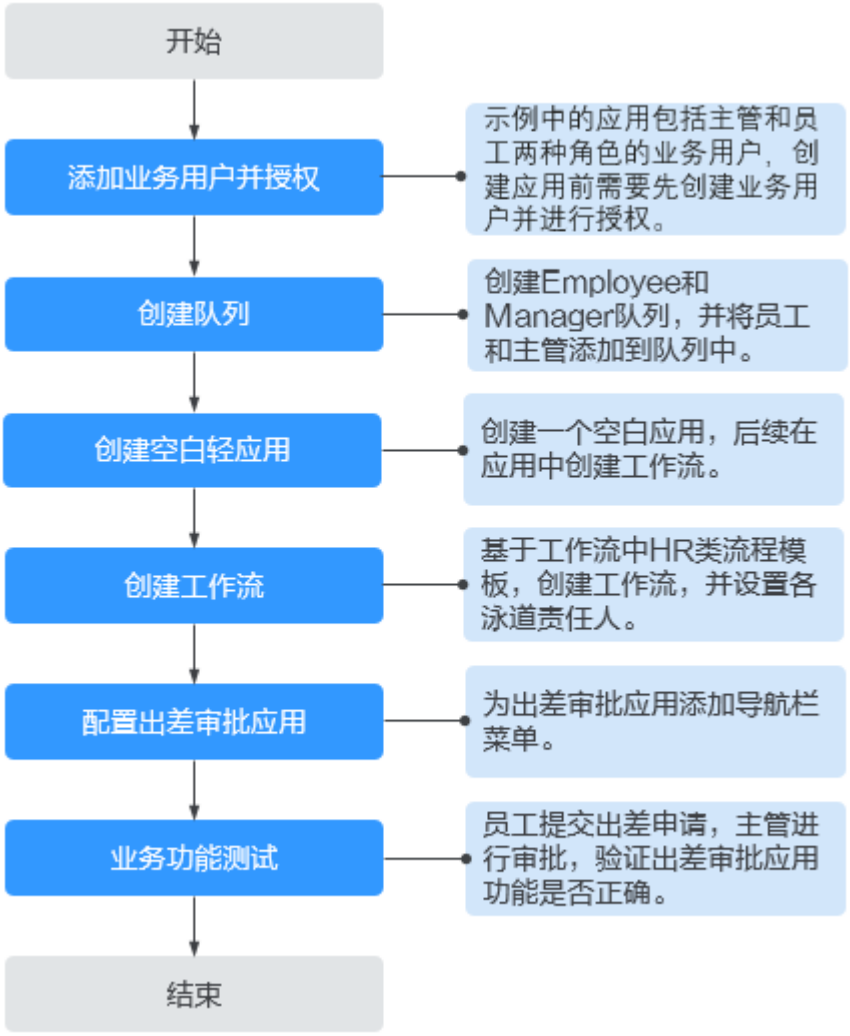
- 基于工作流模板创建出差电子流。
- 发送邮件。

图 6-1 出差审批应用最终效果图

操作流程

在AstroZero中开发出差审批应用的流程，如图6-2所示。

图 6-2 出差审批应用开发流程



1. **步骤一：添加业务用户并进行授权**

   本示例中的应用包括主管和员工两种角色的业务用户。创建出差审批应用前，需要将员工、主管添加到AstroZero中。

2. **步骤二：创建队列**

   队列是AstroZero中的一种成员集，即在实际业务场景中，用来记录一类具有相同权限和任务对象的成员集。本示例中需要创建2个队列，用于在工作流中区分不同角色在流程中处理的任务。

3. **步骤3：创建空白应用**

   创建应用是在AstroZero开发环境开发项目的第一步，也是端到端构建软件应用的入口。

4. **步骤四：创建工作流**

   基于工作流中HR类流程模板，创建工作流，并设置各泳道责任人。

5. **步骤五：配置BTA应用**

   在应用配置中，定义出差审批应用的导航菜单栏。

6. **步骤六：业务功能调测**

验证出差审批流程是否按照预期执行，即员工提交出差申请，主管对申请进行审批，分别测试审批通过、拒绝与重填。

步骤一：添加业务用户并进行授权

本示例中的应用包括主管和员工两种角色的业务用户。创建出差审批应用前，需要将员工、主管添加到AstroZero中。

步骤1 在统一身份认证服务IAM中创建两个子账号（主管和员工）。

1. 以华为账号登录**华为云官网**，在顶部导航栏右侧单击“控制台”，进入华为云控制台。
2. 在左侧导航栏上方，单击，选择服务实例所在的区域项目。
3. 单击，在查找框中搜索“统一身份认证服务”，单击查找到的结果，进入IAM服务控制台。

   您也可以选择“管理与监管 > 统一身份认证服务 IAM”，进入IAM服务控制台。
4. 在“用户”中，单击“创建用户”，创建主管、员工两个用户。

   如何创建用户，请参见**创建IAM用户**。假设，本示例创建主管用户为Helen、员工用户为Mike。

步骤2 在AstroZero中添加子账号并授权。

1. 以华为账号登录华为云。
2. 单击，在查找框中搜索“Astro轻应用”，单击查找的结果，进入AstroZero服务控制台。
3. 在实例页面，单击“进入首页”。
4. 在页面左上角，单击，选择“环境管理 > 环境配置”，进入AstroZero环境配置。

图 6-3 进入 AstroZero 环境配置

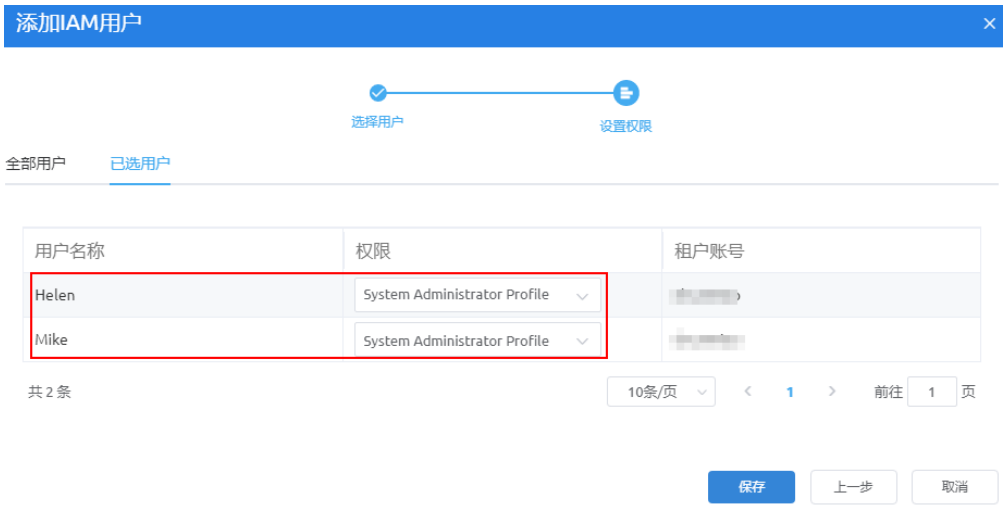
5. 在左侧导航栏中，选择“用户安全 > 用户”，单击“添加IAM用户”。
6. 在全部用户列表中，选择**步骤1**中创建的子账号，单击“下一步”。

图 6-4 选择待添加的子账号

7. 赋予主管和员工System Administrator Profile权限，单击“保存”。

   System Administrator Profile为系统管理员权限，拥有这个权限的用户拥有AstroZero全部权限。在实际的出差审批应用中，只需赋予业务用户Portal User Profile或Anonymous User Profile权限即可。此处为了验证出差审批应用的功能，赋予业务用户（员工Mike、主管Helen）System Administrator Profile权限。如何给业务用户授权，请参见**配置业务用户**。

图 6-5 给子账号授权



----结束

步骤二：创建队列

队列是AstroZero中的一种成员集，即在实际业务场景中，用来记录一类具有相同权限和任务对象的成员集。本示例中应用需要使用到**表6-1**中2个队列，用于在工作流中区分不同角色在流程中处理的任务。

表 6-1 队列信息

名称	说明
Employee	员工队列，添加的队列成员为普通员工，如Mike。
Manager	主管队列（负责一级审批），添加的队列成员为主管，如Helen。

步骤1 以华为账号登录AstroZero环境配置。

步骤2 在主菜单中，选择“维护”。

步骤3 在左侧导航栏中，选择“全局元素 > 队列”。

步骤4 在队列页面，单击“新建”，创建表6-1中的Employee队列。

1. 在新建队列基本信息中，设置队列标签和名称，其他参数保持默认。
   - 标签：新建队列的标签，本示例配置为Employee。
   - 名称：新建队列的名称，本示例配置为Employee。
2. 在队列成员中，单击“添加”，为Employee工作队列添加Mike及当前租户账号为成员。

图 6-6 添加 Mike 成员

3. 单击“保存”，进入队列详情页面。

   在Employee队列详情页面，可以查看新建队列的信息。如果创建队列时未添加成员，在成员信息中单击“添加”，可为队列添加成员。

步骤5 参照步骤4中操作，创建Manager队列。

图 6-7 设置标签和名称

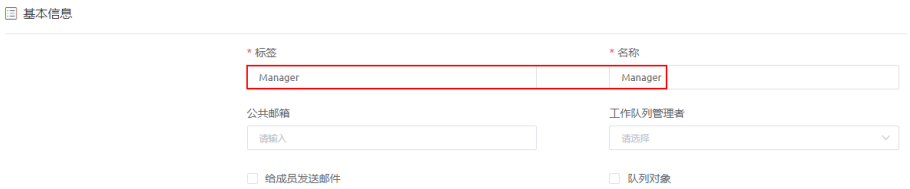


图 6-8 为队列添加 Helen 成员及租户账号

----结束

步骤 3：创建空白应用

创建应用是在AstroZero开发环境开发项目的第一步，也是端到端构建软件应用的入口。

步骤1 以华为账号登录AstroZero服务控制台。

步骤2 在实例列表中，单击“进入首页”，进入应用开发页面。

步骤3 在左侧导航栏中，单击“应用”，进入低代码应用页面。

步骤4 单击新建低代码应用后的，进入新建空白应用页面。

图 6-9 进入创建轻应用入口

步骤5 在新建低代码应用中，选择“标准应用”，单击“确定”。

步骤6 设置应用标签和名称，此处均设置为BTA。

图 6-10 设置轻应用标签和名称

须知

图6-10名称前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改，请谨慎定义。

步骤7 单击“新建”，进入BTA新版应用设计器。

图 6-11 BTA 新版应用设计器

----结束

步骤四：创建工作流

基于工作流中HR类流程模板，创建工作流，并设置各泳道责任人。

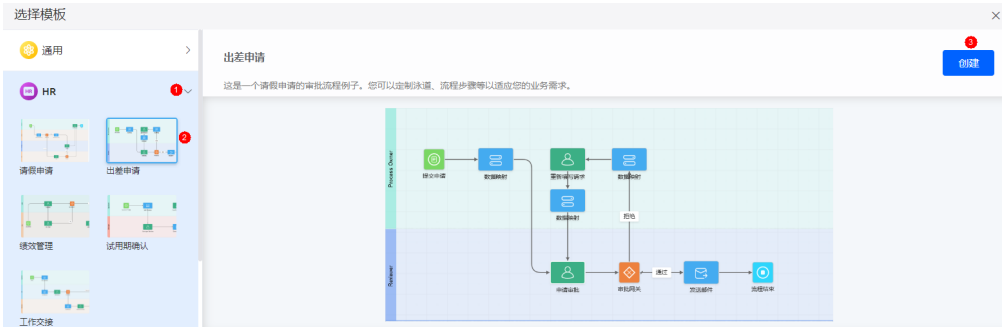
步骤1 创建工作流。

1. 在BTA新版应用设计器的“开始”页面中，单击“新建工作流”。
2. 在新建工作流页面，单击“基于模板”，设置标签、名称和描述信息。

图 6-12 添加工作流

3. 单击“选择模板”，选择“HR”中的“出差申请”模板后，单击“创建”。

图 6-13 选择出差申请模板



创建完成后，自动进入出差申请工作流编辑页面。在设置工作流前，您可以先通过表6-2了解出差申请工作流中各节点的功能。

图 6-14 出差申请工作流开发页面

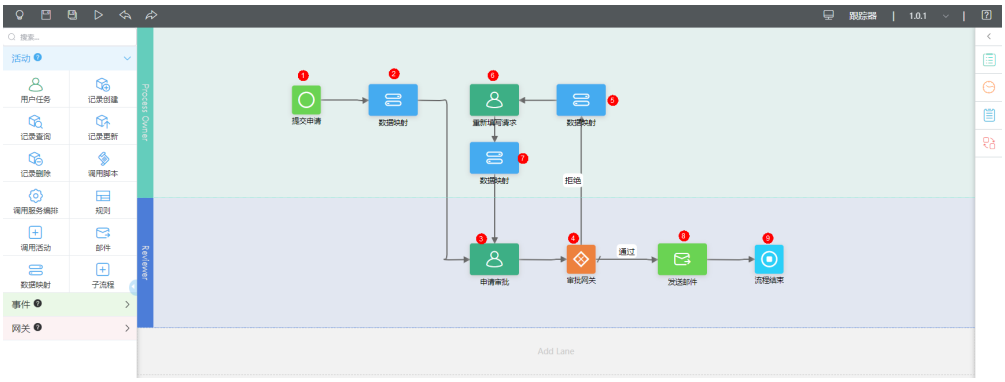


表 6-2 出差申请工作流中各节点功能介绍

编号	节点	功能说明
1	提交申请	开始节点，调用一个标准表单TravelRequest，供出差申请人提交出差申请。
2	数据映射	将出差申请标准表单中的请求字段映射到对象中。
3	申请审批	用户任务，将对象中的字段渲染到标准表单Approve并且确定审批人。
4	审批网关	审批人进行通过或拒绝两种操作。
5	数据映射	申请拒绝后，将Approve标准表单中的字段映射回TravelRequest标准表单。
6	重新填写请求	刷新TravelRequest表单中的内容，重新提交出差申请。

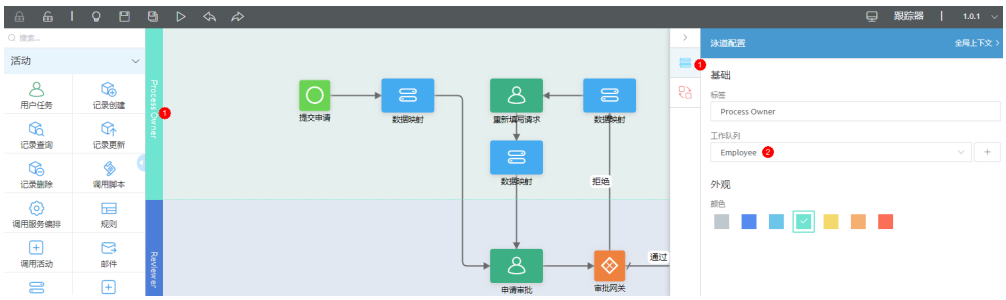
编号	节点	功能说明
7	数据映射	将TravelRequest表单中的字段值映射到Approve中，重新发起审批申请。
8	发送邮件	审批通过，发送邮件将结果告知出差申请人。
9	流程结束	结束节点，执行到此整个工作流执行结束。

步骤2 设置工作流，配置各泳道处理人。

在进行工作流设计前，请先参照**工作流**中内容对工作流设计界面进行全面的了解。

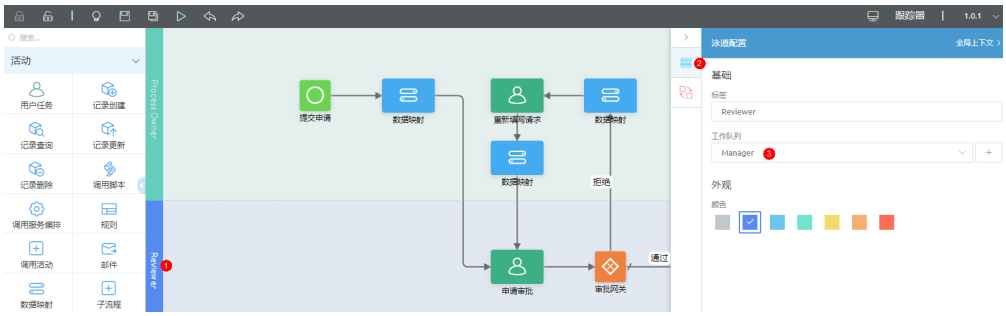
1. 在工作流开发页面，单击“Process Owner”泳道，配置工作队列为“Employee”。

图 6-15 配置员工为申请人



2. 单击“Reviewer”泳道，配置工作队列为“Manager”。

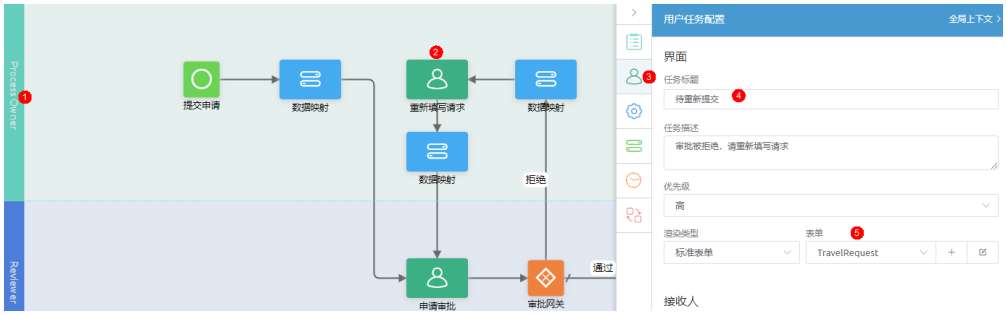
图 6-16 配置主管为审批人



3. 单击泳道“Process Owner”上的“重新填写请求”用户任务元素，设置任务标题为“待重新提交”。

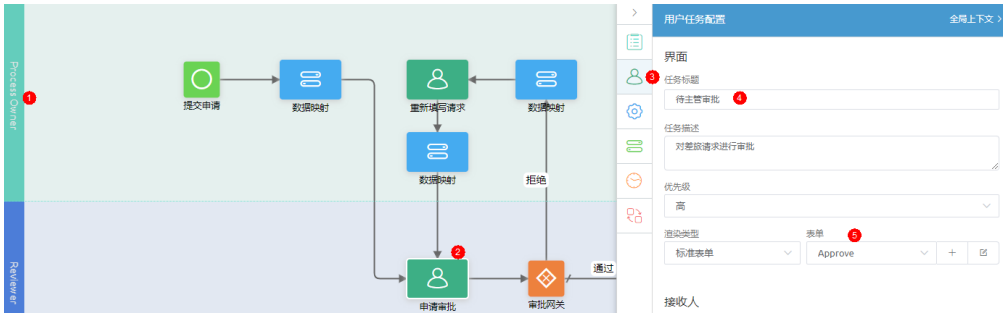
   执行该操作的目的是，后续在“我的待办”中可以看到“待重新提交”的任务状态。

图 6-17 配置为待重新提交



4. 单击泳道“Reviewer”上的“申请审批”用户任务元素，设置任务标题为“待主管审批”。
   执行此操作的目的是，后续在“我的待办”中可以看到“待主管审批”的任务状态。

图 6-18 配置为待主管审批



5. 单击  ，保存工作流。
6. 单击  ，启用工作流。

----结束

步骤五：配置 BTA 应用

在应用配置中，定义出差审批应用的导航菜单栏。

步骤1 在BTA新版应用设计器的“开始”页面，单击底部的“应用导航设置”。

图 6-19 进入应用配置

步骤2 在“主导航设置”页签，单击“Home”，将“菜单名称”设置为“我的待办”，单击“保存”。

图 6-20 编辑页签

步骤3 在主导航设置中，单击“新建”，新建“我要出差”菜单项，单击“保存”。

图 6-21 添加我要出差页签

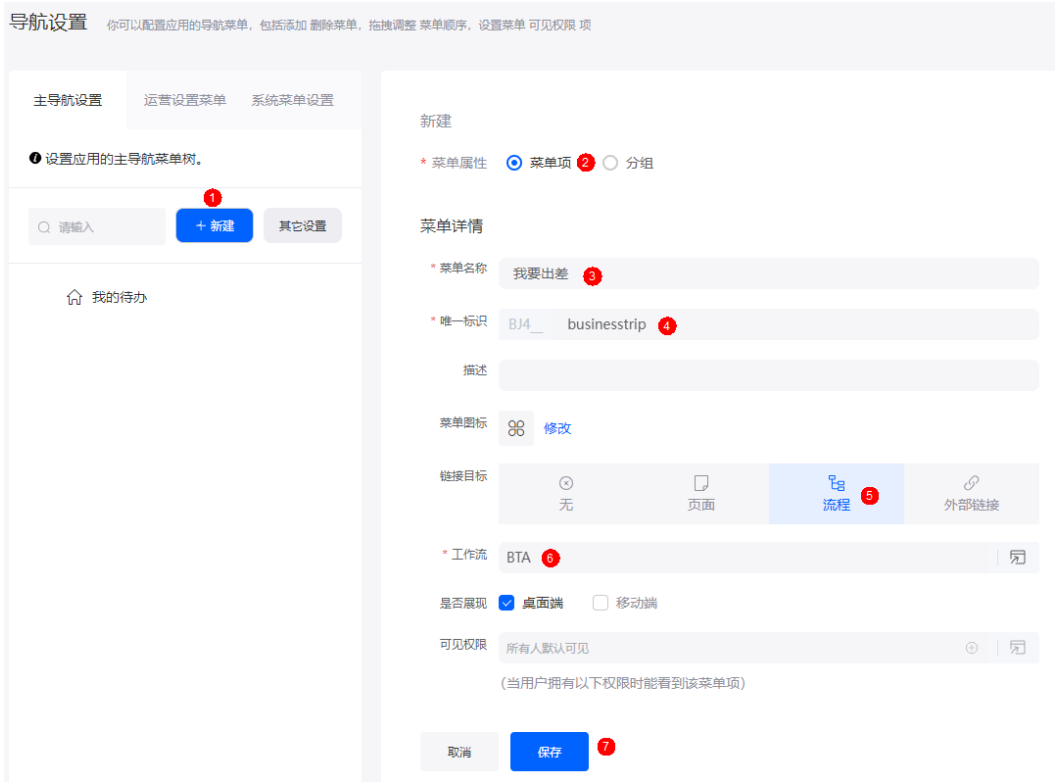


图 6-22 导航条最终效果

步骤4 在BTA新版应用设计器主菜单中，单击“运行 > 立即运行”，可预览出差审批应用。

图 6-23 出差审批应用效果图

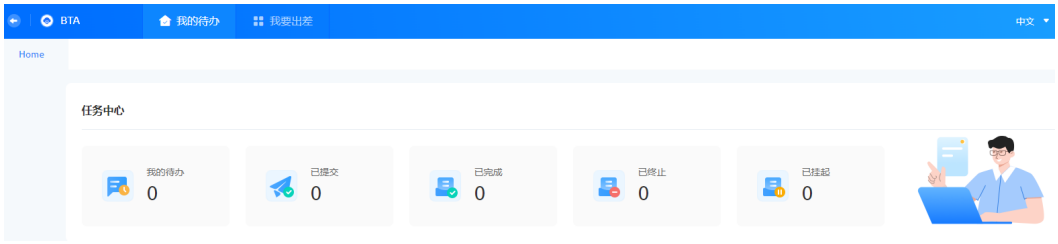
到此，您已完成出差审批应用的开发。

思考：如何将导航栏菜单在出差审批应用顶部显示?

在应用配置页面的“外观设置”中，可以修改菜单样式、应用图标等。

图 6-24 修改菜单样式、应用图标

图 6-25 设置后效果



----结束

步骤六：业务功能调测

出差审批测试流程：员工提交出差申请，主管对申请进行审批，分别测试审批通过、拒绝与重填。

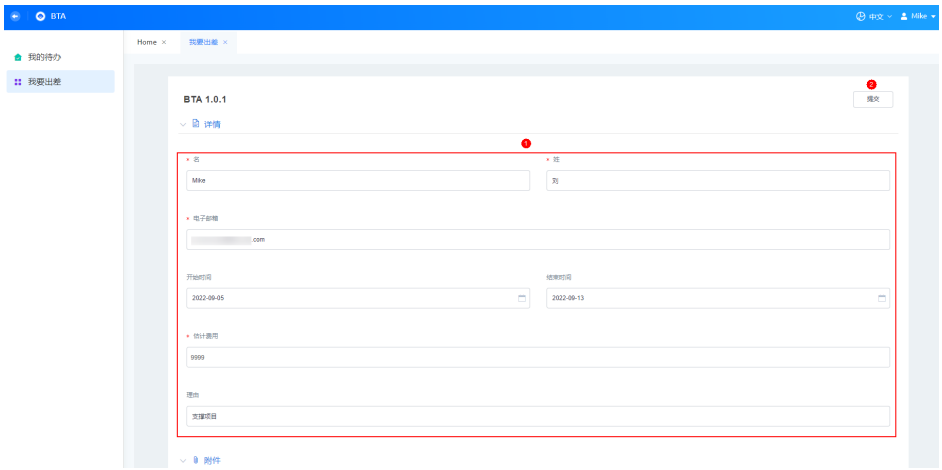
步骤1 我是员工，填写出差申请。

1. 以员工账号（本示例为Mike），登录AstroZero服务控制台。

图 6-26 员工登录 AstroZero

2. 在实例页面，单击“进入首页”。
3. 在左侧导航栏中，单击“应用”，进入低代码应用页面。
4. 在低代码应用列表中，单击已创建BTA应用后的“编辑”，进入BTA应用设计器。
5. 在主菜单中，单击“运行 > 立即运行”，进入出差审批应用预览页面。
6. 单击“我要出差”，填写出差信息，单击“提交”。

图 6-27 员工填写出差申请



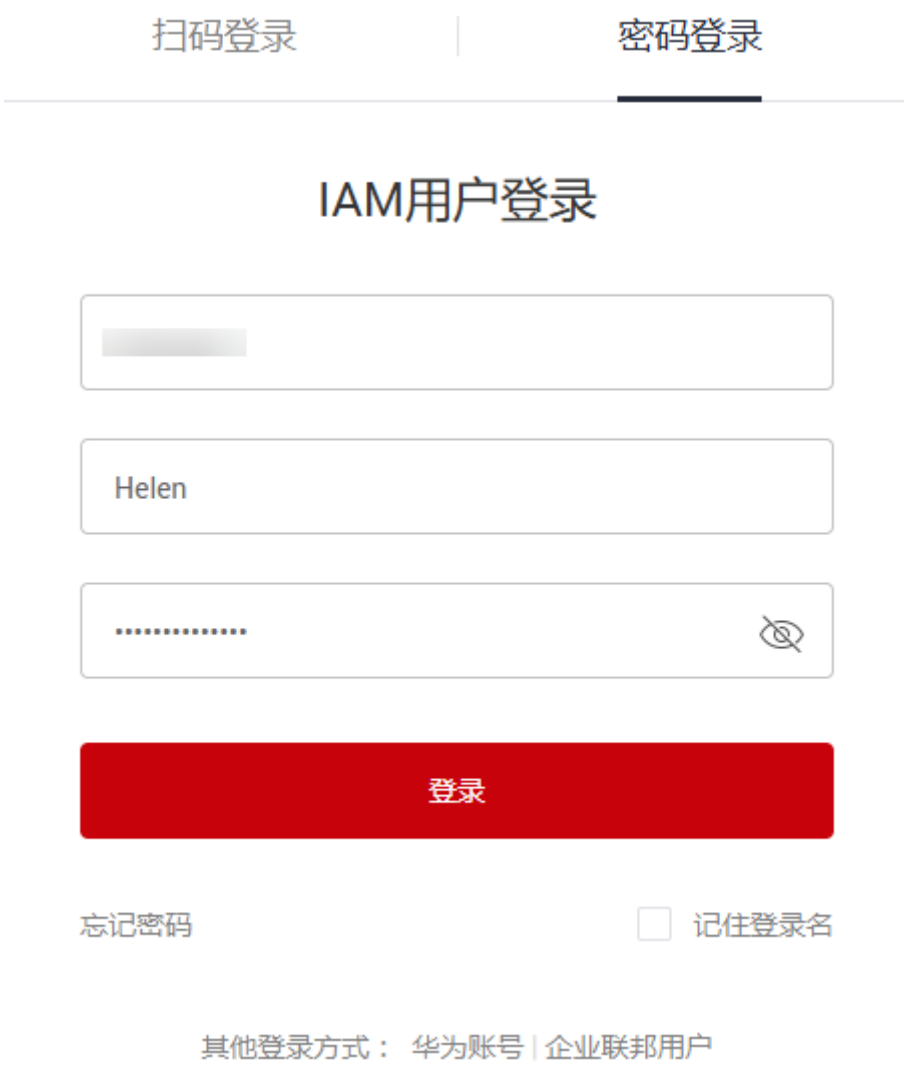
7. 在“我的待办 > 我的申请”中，可查看到已提交的出差申请电子流。

图 6-28 查看我的申请

步骤2 我是主管，审批出差申请。

1. 以主管账号（本示例为Helen），登录AstroZero服务控制台。

图 6-29 主管登录 AstroZero

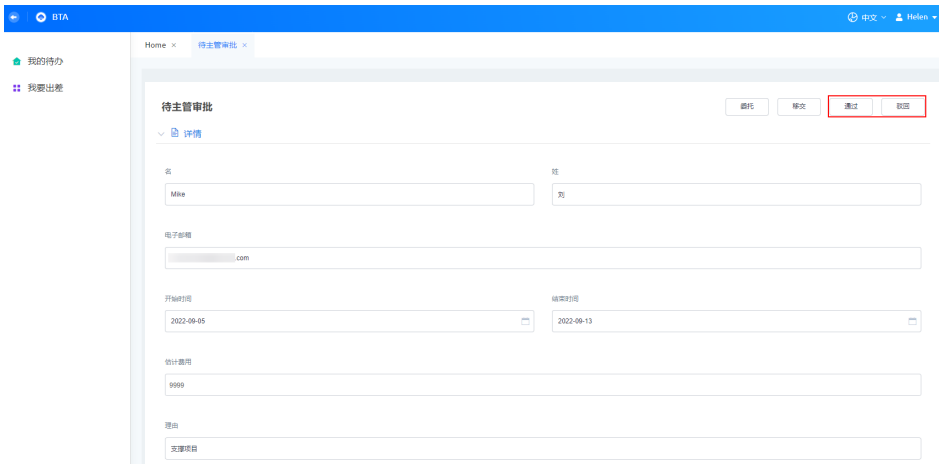


2. 在实例页面，单击“进入首页”。

3. 在左侧导航栏中，单击“应用”，进入低代码应用页面。
4. 在低代码应用列表中，单击已创建BTA应用后的“编辑”，进入BTA应用设计器。
5. 在主菜单中，单击“运行 > 立即运行”，进入出差审批应用预览页面。
6. 单击“我的待办”，在我的待办列表中，单击“待主管审批”，进入主管审批页面。

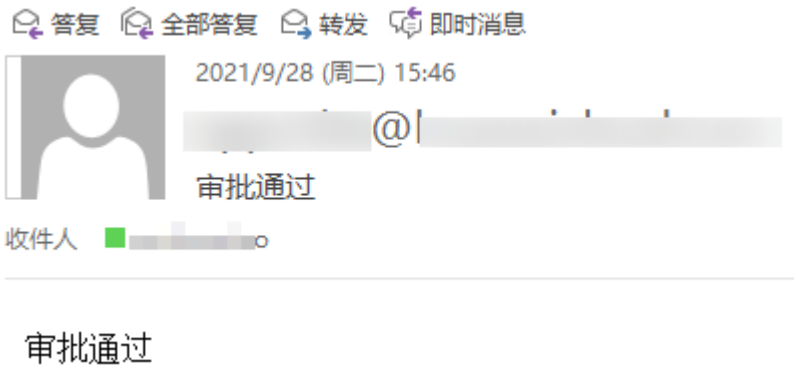
图 6-30 我的待办

图 6-31 主管审批页面



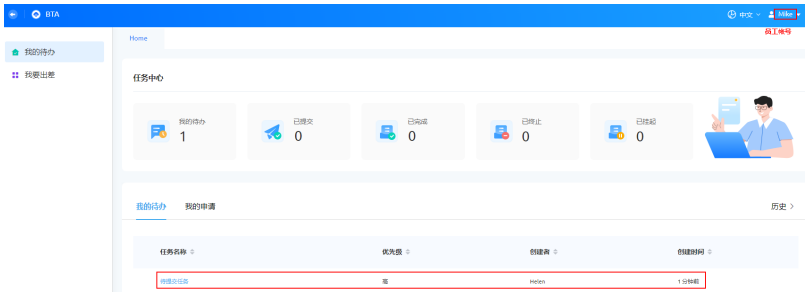
7. 单击“通过”或“驳回”。
   - 如果主管通过了审批，会发送审批通过的邮件到图6-27中的员工电子邮箱中，如图6-32。

图 6-32 审批通过邮件



- 如果主管驳回了审批，在员工“我的待办”中，可查看到被主管退回的申请记录，如**图6-33**。

图 6-33 查看被拒绝的记录



恭喜您，完成了出差审批应用的开发和功能测试。通过本章节的操作，相信您对AstroZero中的轻应用及工作流有了一定的了解。

说明

思考：如何将出差审批应用发布到运行环境？发布到运行环境后，队列中数据会同步到运行环境中吗？

- **问题1**：免费版不提供运行环境，如果需将应用发布到运行环境，请购买专享版或专业版本，详情可参见**购买AstroZero商用实例**。如何编译发布应用，请参见**如何一键部署应用**。
- **问题2**：出差审批应用发布到运行环境后，开发环境中的队列信息不会同步到运行环境中，请在运行环境中重新执行**步骤二：创建队列**。

----结束

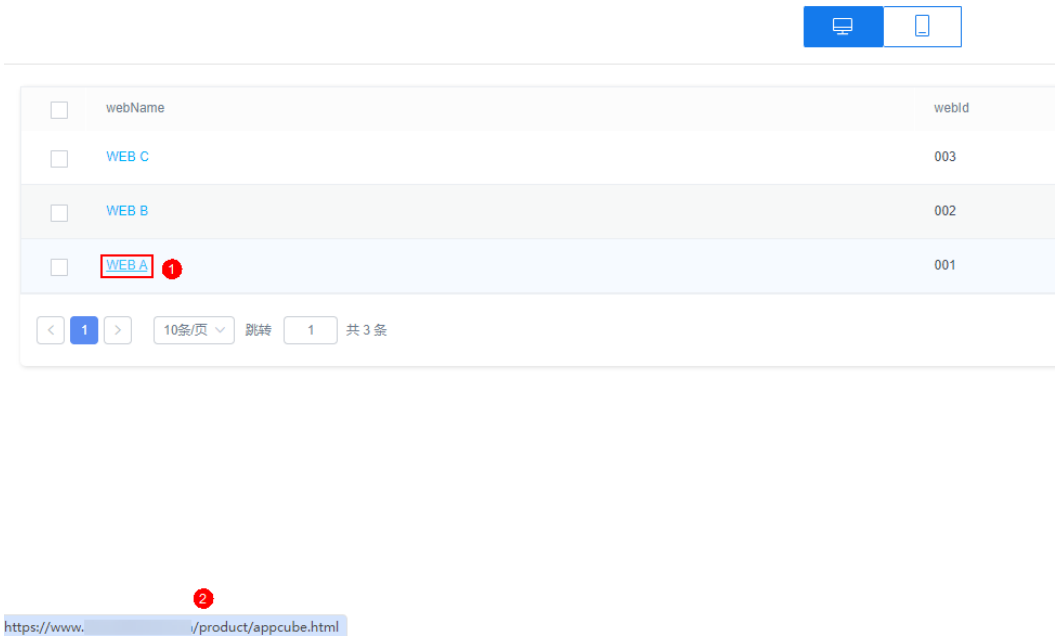
7 标准页面专项

7.1 为 AstroZero 标准页面中表格的数据增加链接

期望实现效果

在标准页面中，支持为表格中的数据增加超链接，来提升用户体验和数据交互的便捷性。例如，在表格的webName列中，将鼠标移动至WEB A上在页面的左下角可查看到对应的链接地址，单击会跳转到对应的页面。

图 7-1 实现效果



功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。

2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 7-2 创建一个空白应用

表 7-1 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建对象“websiteList”，并为对象添加字段。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“websiteList”，单击“确定”。

图 7-3 新建对象 websiteList

表 7-2 新建 websiteList 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	websiteList
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	websiteList

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加webName字段。

图 7-4 添加 webName 字段

表 7-3 添加 webName 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	webName
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	webName
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

5. 按照上述操作，为对象添加webLink和webId字段。

表 7-4 添加 webLink 和 webId 字段

显示名称	唯一标识	字段类型
webLink	webLink	文本区

显示名称	唯一标识	字段类型
webId	webId	文本

6. 在“数据”页签，单击“添加”，为对象添加图7-5中数据。

图 7-5 为对象添加数据

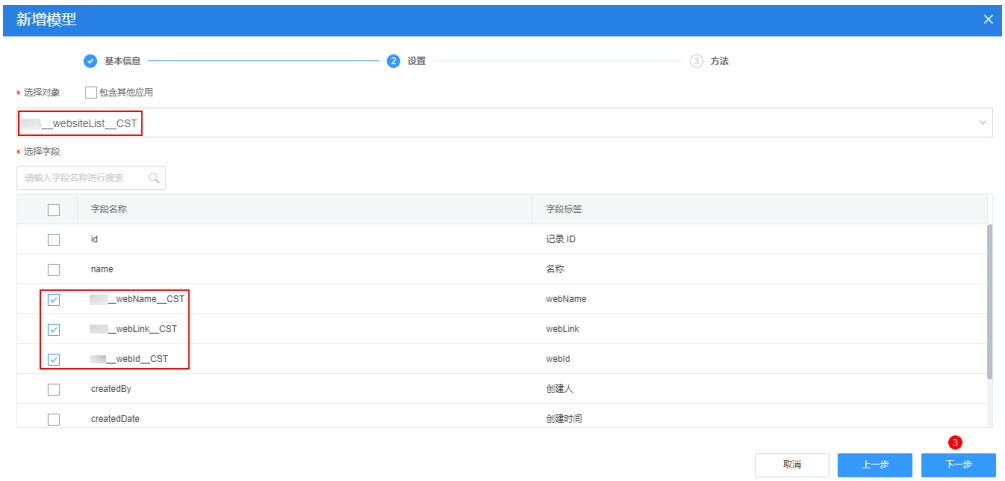
步骤3 新建对象模型。

1. 在应用设计器中，选择“界面”，单击页面后的“+”。
2. 输入页面的标签和名称，单击“添加”，新建一个标准页面。
3. 在标准页面底部，单击“模型视图”，将页面从设计视图切换到模型视图。
4. 单击“新增模型”，输入模型名称（link）、“来源”选择“对象”，单击“下一步”。

图 7-6 新建模型

5. 选择步骤3中创建的对象和添加的字段，单击“下一步”，再单击“确定”，完成模型的创建。

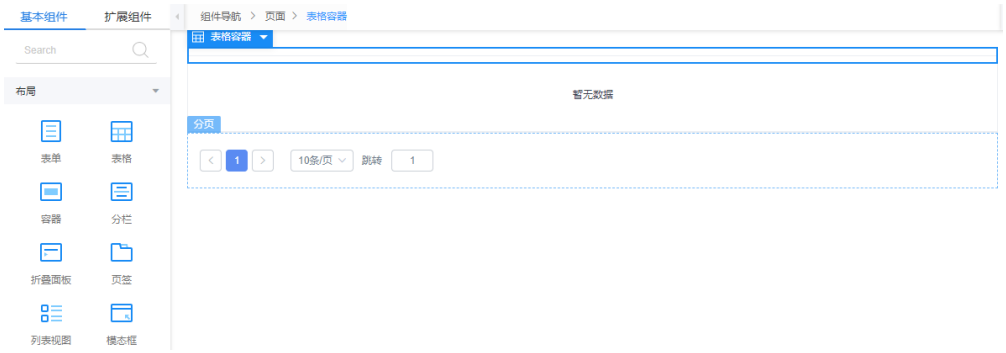
图 7-7 选择对象和字段



步骤4 返回设计视图页面，新建表格关联模型。

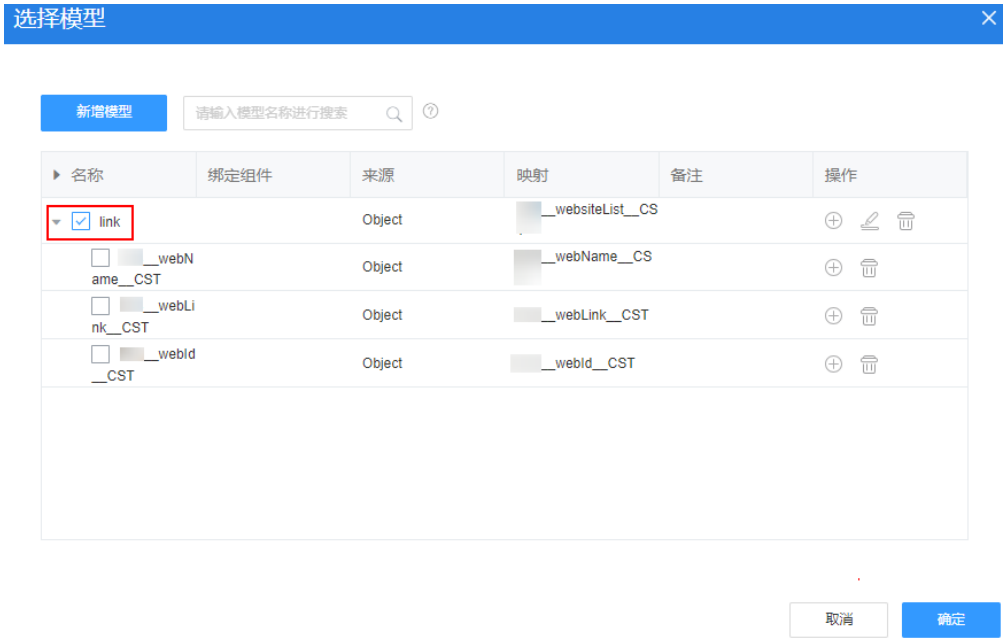
1. 在标准页面中拖入一个表格组件。

图 7-8 拖入表格组件



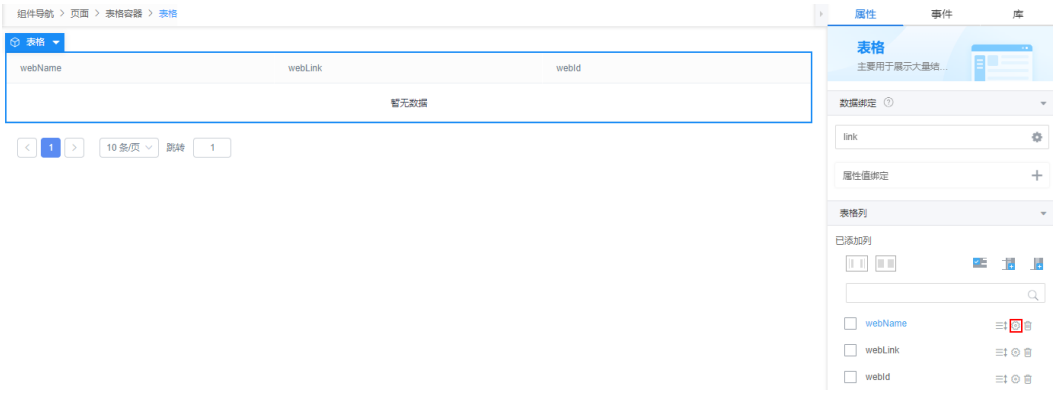
2. 选中表格组件，在“属性 > 数据绑定 > 值绑定”中，单击 。
3. 选中**步骤3**中创建的模型，单击“确定”。

图 7-9 选择模型



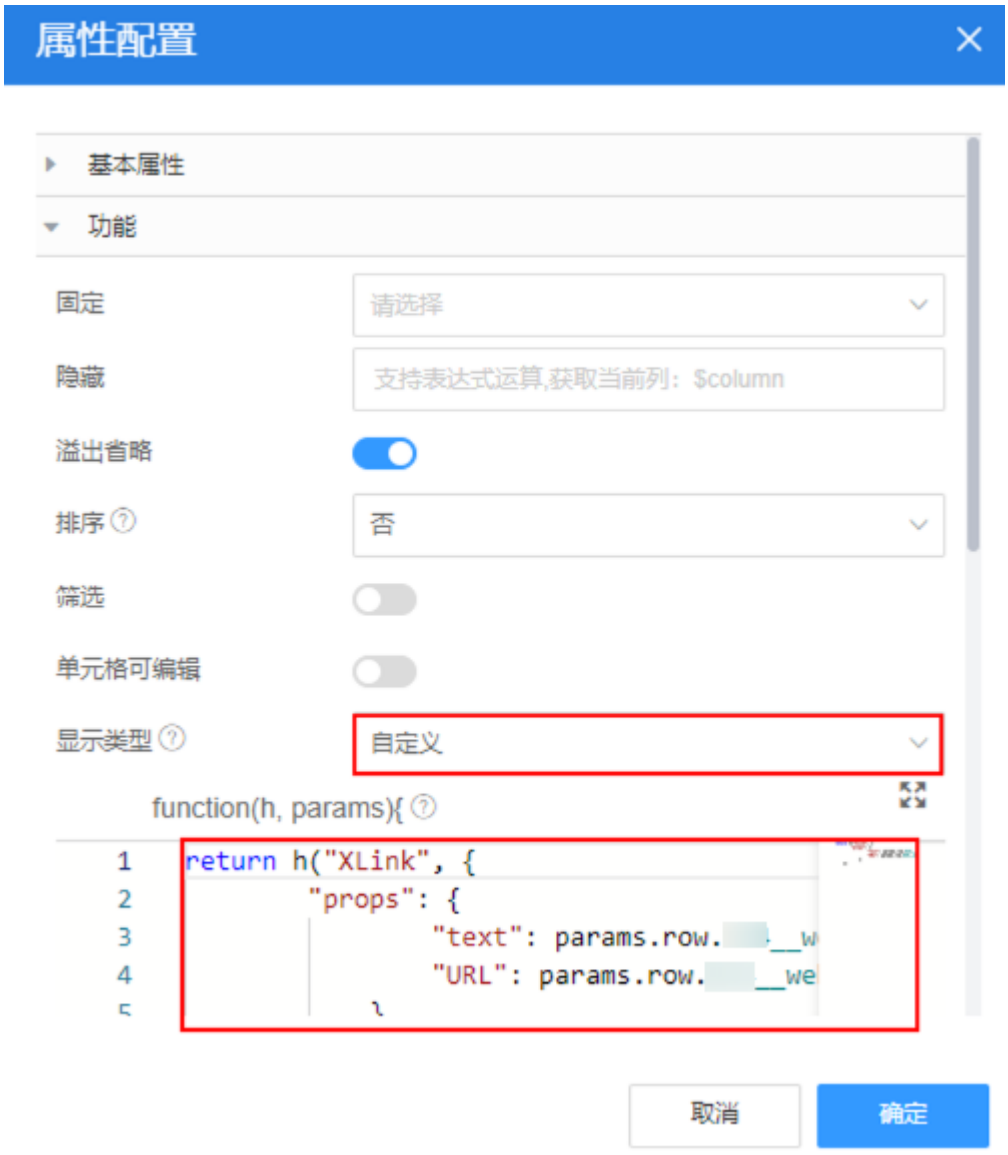
步骤5 在“表格列”中，单击webName列后的 。

图 7-10 选择所需的列



步骤6 在“属性配置 > 功能”中，为webName列添加URL链接信息。

图 7-11 设置列属性



将“显示类型”设置为“自定义”，并输入如下内容：

```
return h("XLink", {
        "props": {
                "text": params.row.命名空间__webName__CST,
                "URL": params.row.命名空间__webLink__CST
            }
    })
```

步骤7 单击页面上方的，保存页面。

步骤8 保存成功后，单击页面上方的，预览效果。

----结束

7.2 为 AstroZero 标准页面中的表格增加求和等计算能力

期望实现效果

在标准页面中，支持为表格增加求和、求积等计算能力，来提升数据的处理效率。例如，将表格中“商品花费”列的值设置为 “商品数*价格+其他成本”。

图 7-12 实现效果

功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 7-13 创建一个空白应用

表 7-5 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建对象“ProductList”，并为对象添加字段和数据。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“ProductList”，单击“确定”。

图 7-14 创建对象 ProductList

表 7-6 新建 ProductList 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	ProductList
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	ProductList

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加productName字段。

图 7-15 添加 productName 字段

表 7-7 添加 cost 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	商品名称
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	productName
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

5. 按照上述操作，为对象添加表7-8中字段。

表 7-8 添加其他字段

显示名称	唯一标识	字段类型
商品数目	productNum	数字

显示名称	唯一标识	字段类型
商品价格	productPrice	数字
其他成本	cost	数字

图 7-16 查看添加的字段

6. 选择“数据”页签，单击“添加”，为对象添加图7-17中数据。

图 7-17 为对象添加数据

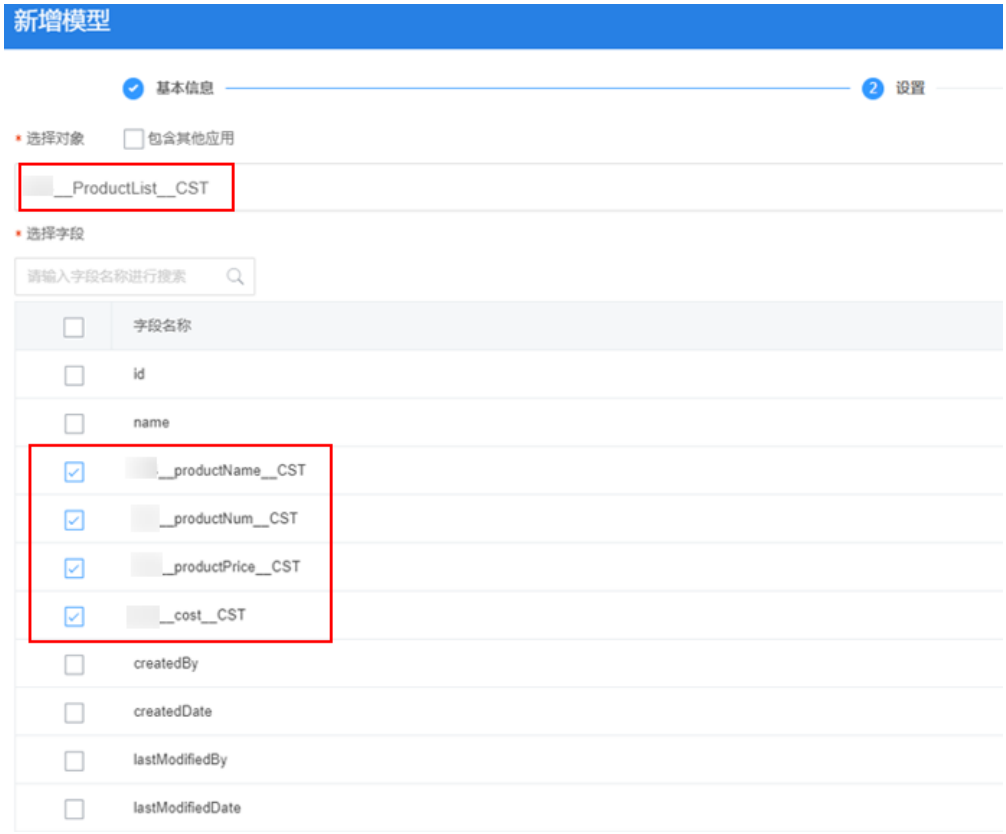
步骤3 新建对象模型。

1. 在应用设计器中，选择“界面”，单击页面后的“+”。
2. 输入页面的标签和名称，单击“添加”，新建一个标准页面。
3. 在标准页面底部，单击“模型视图”，将页面从设计视图切换到模型视图。
4. 单击“新增模型”，输入模型名称（如productCost）、“来源”选择“对象”，单击“下一步”。

图 7-18 新建模型

5. 选择**步骤3**中创建的对象和添加的字段，单击“下一步”，再单击“确定”，完成模型的创建。

图 7-19 选择对象和字段



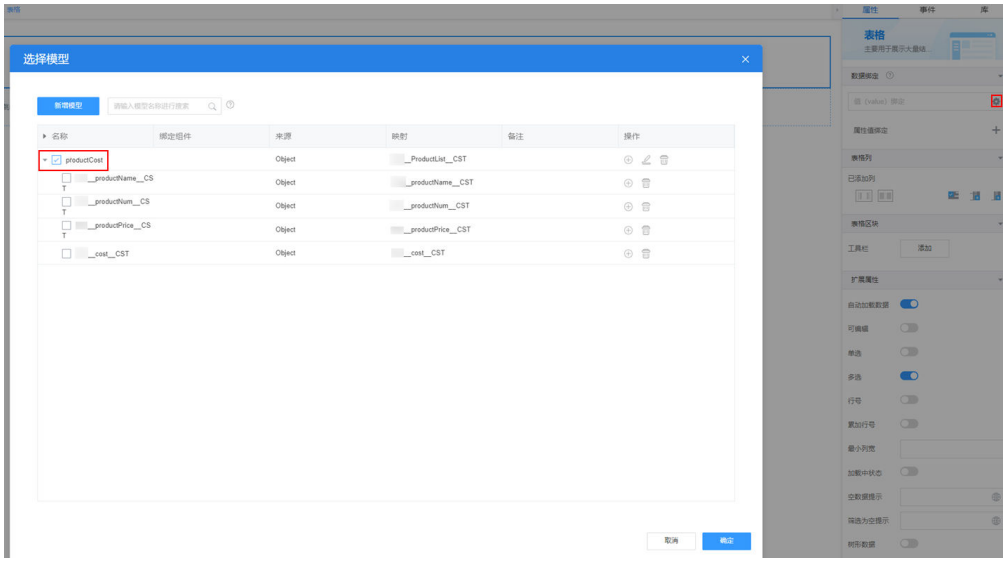
步骤4 返回设计视图页面，新建表格关联模型。

1. 在标准页面中拖入一个表格组件。

图 7-20 拖入表格组件

2. 选中表格组件，在“属性 > 数据绑定 > 值绑定”中，单击 。
3. 选中**步骤3**中创建的模型，单击“确定”。

图 7-21 选择模型



步骤5 选中表格，在“属性 > 表格列 > 已添加列”中，单击 ，添加一个空白列。

图 7-22 添加一个空白列

步骤6 单击新增空白列后的 ，修改“列名称”为“商品花费”，将“显示类型”设置为“自定义”并输入自定义代码。

图 7-23 修改列标题为商品花费

图 7-24 显示类型选择自定义并输入自定义代码

自定义代码内容如下：

```
return h("XInputNumber", {
      "props": {
          "value": params.row.命名空间__productPrice__CST * params.row.命名空间__productNum__CST + params.row.命名空间__cost__CST,
          "readonly": true
      }
  })
```

步骤7 单击页面上方的，保存页面。

步骤8 保存成功后，单击页面上方的，预览效果。

----结束

7.3 为 AstroZero 调查问卷应用新增调查项

期望实现效果

调查问卷页面中的问卷项由对象模型定义，如果需要添加或修改调查项，需要先修改对象模型“问卷记录表”。例如，在Astro轻应用开发者调查问卷应用中，新增调查项“您最常使用的功能或者您最感兴趣功能有哪些？”（效果如图7-25），需要在问卷记录表对象模型中添加自定义字段。

图 7-25 预览配置效果

功能实现方法

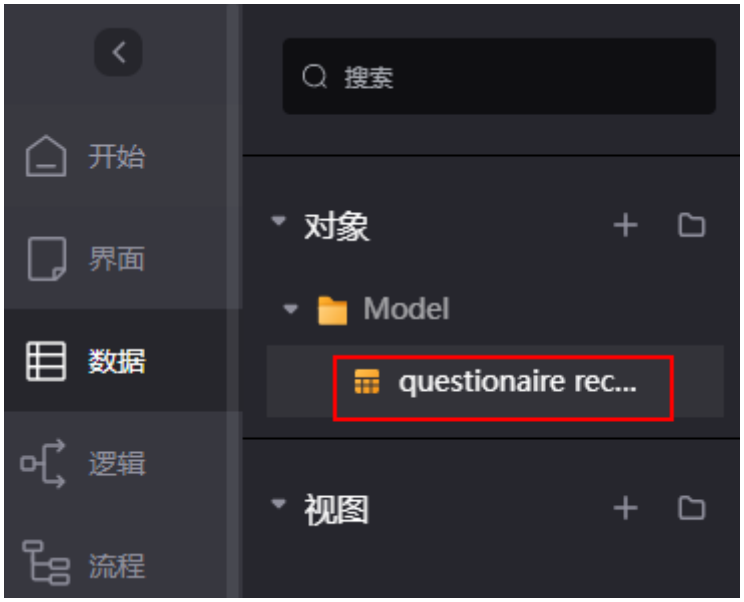
步骤1 参考使用AstroZero低代码预置模板创建调查问卷应用中操作，安装Astro轻应用调查问卷应用。

步骤2 自定义调查问卷项，即在问卷记录表对象模型中添加字段。

新建一个自定义字段，后续在调查问卷页面的数据模型中需要关联该字段。

1. 在调查问卷应用设计器的左侧导航栏中，选择“数据”。
2. 在“对象 > Model”目录下，单击“questionaire record”。

图 7-26 单击“questionaire record”对象



3. 在对象中，单击，进入对象详情页。
4. 在字段页签中，单击“添加”，进入添加字段页面。

图 7-27 单击添加按钮

5. 设置新增字段的显示名称、唯一标识和字段类型，单击“确认”。

图 7-28 新增 questionCode10 字段

- 显示名称：新建字段在页面显示的名称，长度不能超过63个字符。本示例配置为“questionCode10”。
- 唯一标识：新建字段在系统中的唯一标识，创建后不支持修改，单击输入框自动生成，本示例配置为“questionCode10”。

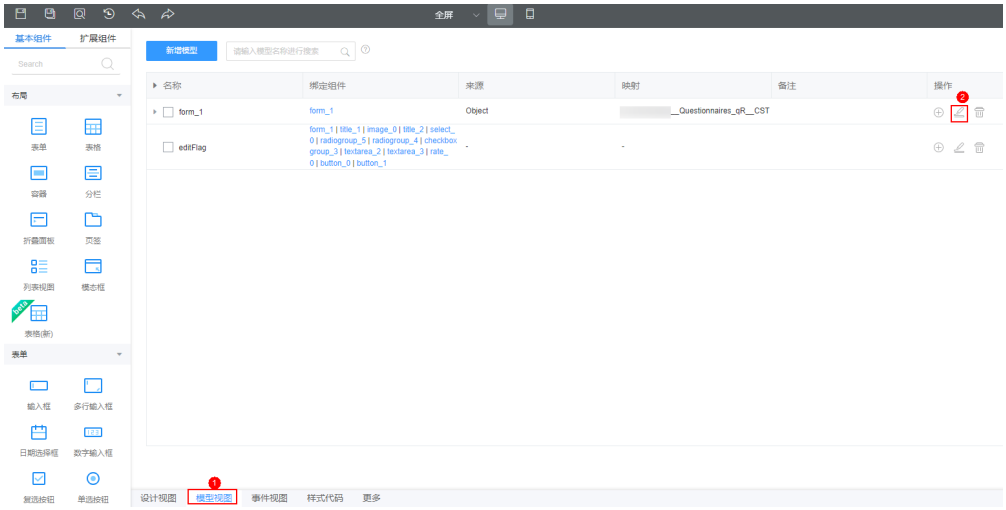
  标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。
- 字段类型：新增字段的类型，本示例配置为“文本区”。

步骤3 自定义调查问卷页面。

AstroZero的标准页面通过数据模型驱动，页面所有的逻辑都围绕数据模型展开。在将模型与前台页面组件（例如输入框）或者后台逻辑绑定后，您只需要关注模型数据的实例化和处理，不需要关注页面的渲染和展示。

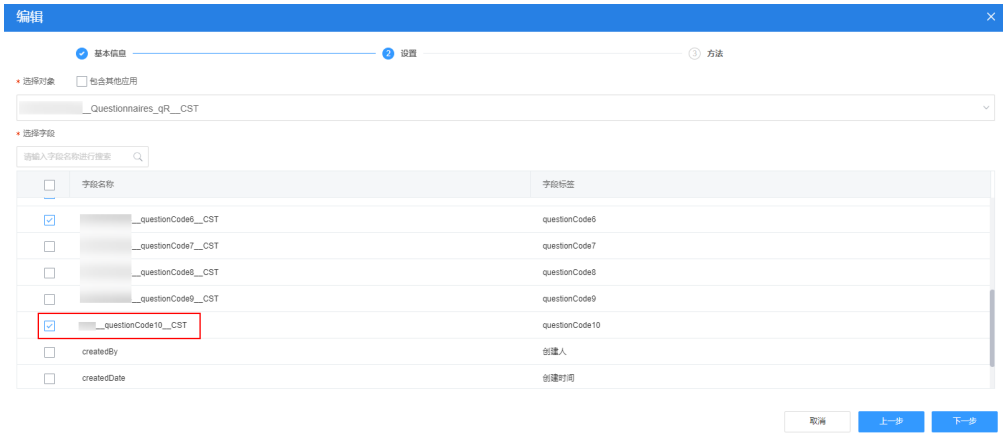
1. 在调查问卷应用设计器的左侧导航栏中，选择“界面”。
2. 在“页面 > Page”目录下，单击用户体验页面。
3. 选择标准页面编辑器下方的“模型视图”，从“设计视图”切换到“模型视图”，单击“form_1”后的 。

图 7-29 模型视图



4. 勾选**步骤2**中添加的对象字段（*命名空间*__questionCode10__CST），单击“下一步”。

图 7-30 勾选字段



5. 单击“确定”，返回模型视图页面。
6. 选择标准页面编辑器下方的“设计视图”，从“模型视图”切换到“设计视图”。
7. 从左侧基本组件中，拖拽“下拉框”组件到“1. 您的工作类型？”下方。

图 7-31 拖入下拉框

8. 选中下拉框组件，参照**表7-9**设置组件的属性。

图 7-32 绑定数据

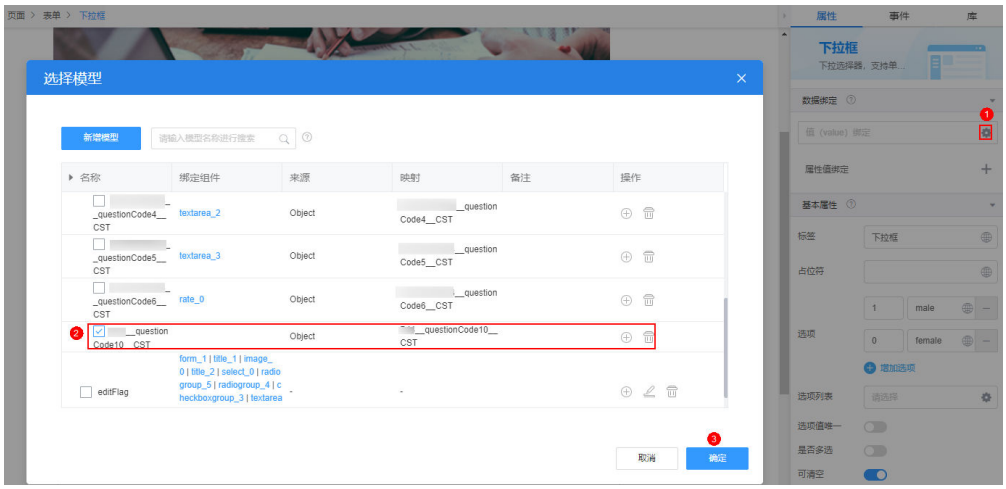


图 7-33 设置属性

图 7-34 设置标签

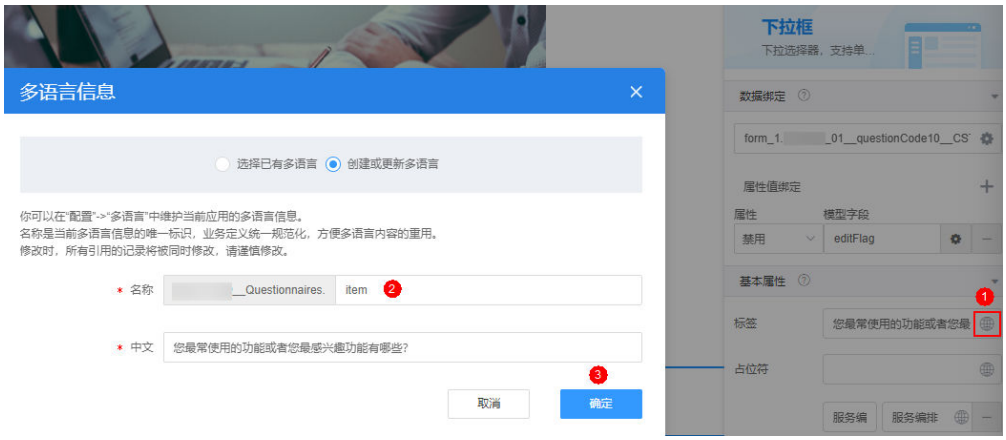


表 7-9 下拉框属性说明

参数	说明
数据绑定	如图7-32所示，设置组件绑定的数据模型，在下拉框“属性”页签单击数据绑定后的图标，选择步骤2中添加的模型字段。 设置属性值绑定，在属性值后单击“+”，添加“禁用”属性，模型字段选择“editFlag”。目的是问卷填写完成后查看结果时，不可编辑。
标签	下拉框的显示名称，只有下拉框放在表单form中才生效。 支持国际化配置，在配置该属性时，可选择已有多语言、创建或更新多语言。此处创建的多语言会保存在租户的多语言库中。 如图7-34所示，单击标签后的，选择“创建或更新多语言”，设置多语言名称和多语言内容，例如“名称”设置为“item”，中文设置为“您最常使用的功能或者您最感兴趣的功能有哪些？”。
选项	下拉框单击后可选择的选项内容。 设置如图7-33所示选项。其中选项值支持国际化配置，在配置该属性时，选择“创建或更新多语言”，设置多语言名称和多语言内容。
是否多选	配置下拉框是否提供多选的能力。 本示例选择打开此开关。

9. 单击页面上方的，保存页面。
10. 保存成功后，单击页面上方的，查看页面配置效果。

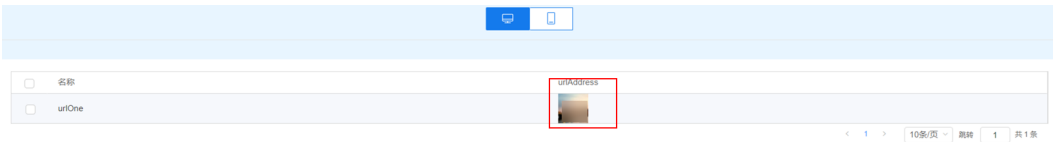
----结束

7.4 在 AstroZero 标准页面的表格中显示图片

期望实现效果

在标准页面中，通过自定义列的显示类型，可以将图片显示在表格中。在表格中显示图片可增强信息的表达效果，信息更直观、生动和易于理解。

图 7-35 实现效果



功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 7-36 创建一个空白应用

表 7-10 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 创建对象“urlList”，并为对象添加字段。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“urlList”，单击“确定”。

图 7-37 创建对象 urlList

表 7-11 新建 urlList 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	urlList

参数	说明	示例
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	urlList

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加urlAddress字段。

图 7-38 添加 urlAddress 字段

表 7-12 添加 urlAddress 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	urlAddress

参数	说明	示例
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	urlAddress
字段类型	单击***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本区

5. 在“数据”页签，单击“添加”，为对象添加数据。
   其中，“urlAddress”的值请配置为图片的存放地址。

图 7-39 为对象添加数据

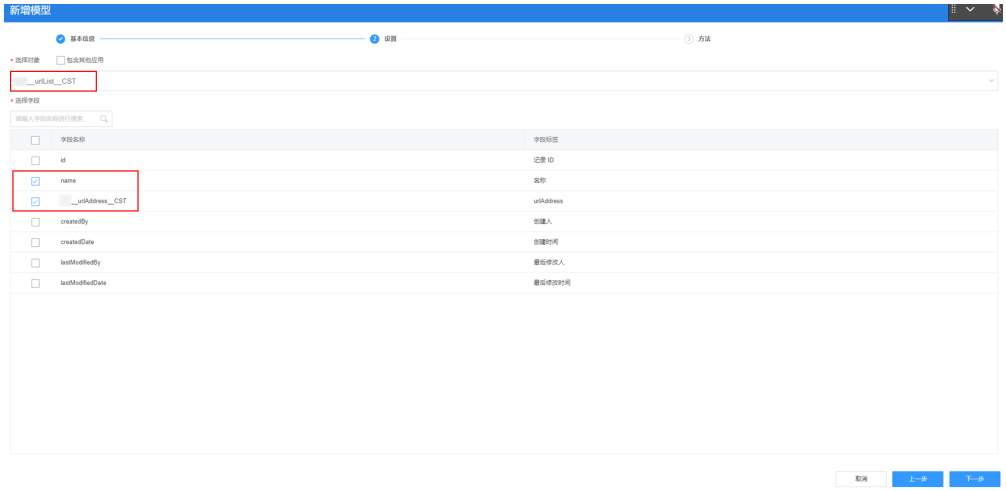
步骤3 新建对象模型。

1. 在应用设计器中，选择“界面”，单击页面后的“+”，新建一个标准页面。
2. 在标准页面底部，单击“模型视图”，从设计视图切换到模型视图。
3. 单击“新增模型”，输入模型名称（如urlMod）、“来源”选择“对象”，单击“下一步”。

图 7-40 新建模型

4. 选择步骤3中创建的对象和添加的字段，单击“下一步”。

图 7-41 选择对象和字段



5. 直接单击“确定”，完成模型创建。

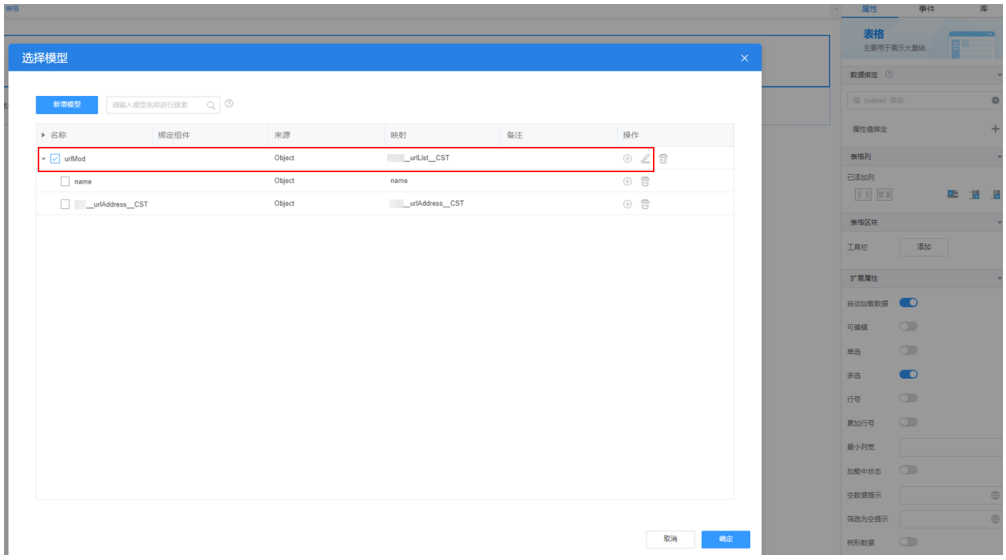
步骤4 返回设计视图页面，新建表格关联模型。

1. 在标准页面底部，单击“设计视图”，从模型视图切换到设计视图。
2. 在标准页面中拖入一个表格组件。

图 7-42 拖入表格组件

3. 选中表格组件，在“属性 > 数据绑定 > 值绑定”中，单击 。
4. 选中**步骤3**中创建的模型，单击“确定”。

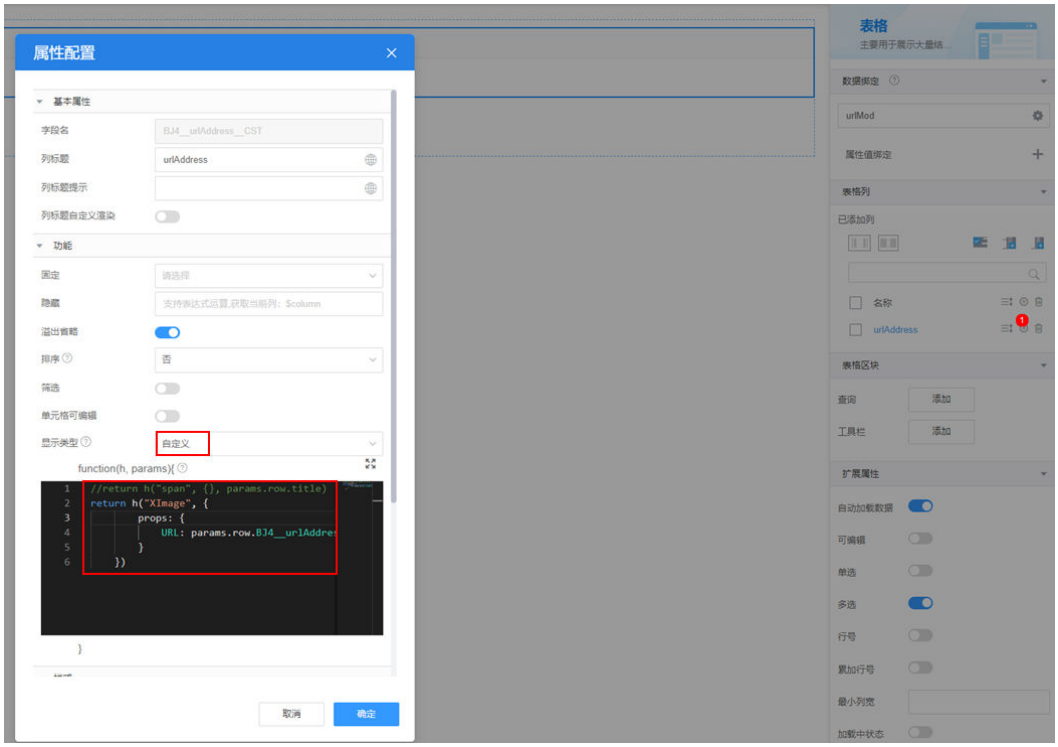
图 7-43 选择模型



步骤5 选中表格，在“属性 > 表格列 > 已添加列”中，单击待显示图片列（urlAddress）后的 。

步骤6 设置属性，单击“确定”，返回标准页面。

图 7-44 设置属性



将“显示类型”设置为“自定义”，并输入如下内容。

```
return h("XImage", {
        "props": {
          "URL": params.row.命名空间__urlAddress__CST, width: 50, height: 50
```



```
    }
})
```

步骤7 单击页面上方的，保存页面。

步骤8 保存成功后，单击页面上方的，查看页面配置效果。

----结束

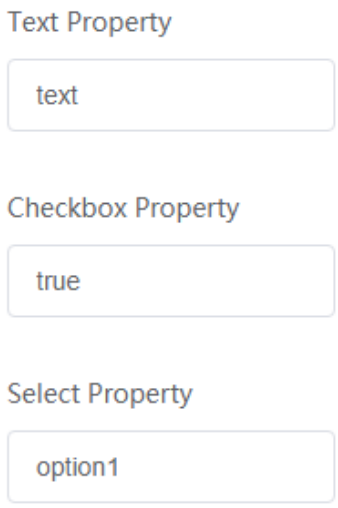
8 高级页面专项

8.1 使用 AstroZero 自定义组件在页面中的属性

期望实现效果

组件预置的属性不能满足您的业务需求时，支持为组件自定义属性。例如，为组件widget_demo_property自定义Text Property、Checkbox Property和Select Property三个属性。组件中自定义属性在页面中的最终呈现效果如图8-1所示。

图 8-1 自定义组件属性



功能实现方法

步骤1 下载组件模板。

1. 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。

2. 单击，选择“环境管理 > 环境配置”，进入环境配置。
3. 在主菜单中，选择“维护”。
4. 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 组件模板”。
5. 在组件列表中，单击“widgetPropertyTemplate”，进入模板详情页。
6. 单击“下载”，设置组件的名称为“widget_demo_property”，单击“保存”，将模板下载到本地。
   如果选择“下载原始模板”，下载到本地的组件包中，组件名称不会被修改。

图 8-2 保存模板

步骤2 自定义组件属性。

1. 在“widget_demo_property.editor.js”的“propertiesConfig”中，定义Widget属性，包含属性的类型、名称和在界面展示的标签名。
   如下加粗代码所示，“widget_demo_property.editor.js”中分别定义了text、checkbox以及select类型的三个属性参数。

```
widget_demo_property = widget_demo_property.extend({
    /
    Config to define Widget Properties
     */
    propertiesConfig:[{
        config: [{
                "type": "text",
                "name": "textProperty",
                "label": "Text Property",
                "value": "text"
            },
            {
                "type": "checkbox",
                "name": "checkboxProperty",
                "label": "Checkbox Property",
                "value": "true"
            },
            {
                "type": "select",
                "name": "selectProperty",
                "label": "Select Property",
                "options": [{
                        "label": "option1",
                        "value": "option1",
                        "selected": "true"
                    },
                    {
                        "label": "option2",
                        "value": "option2"
                    }
                ]
```



```
            }
        ]
    }],

    /
    Triggered when the user Creates a new widget and used to initialize the widget properties
     */
    create : function(cbk)
    {
        if(cbk)
        {
            this._super();
            cbk();
        }
    }
});

var params = {};
Studio.registerWidget("widget_demo_property", "widget_demo_property", params);
```

其中：

- type：属性的类型。
- name：属性的名称。
- label：属性在界面展示的标签名。
- value：属性的默认取值。如果属性是select类型，则需要定义选项“options”。

2. 在“widget_demo_property.js”中，通过定义widgetProperties变量“var widgetProperties = thisObj.getProperties();”，调用“thisObj.**getProperties**”方法，来获取到这些属性。

```
var widget_demo_property = StudioWidgetWrapper.extend({
    /
    Triggered when initializing a widget and will have the code that invokes rendering of the widget
     */
    init : function()
    {
        var thisObj = this;
        thisObj._super.apply(thisObj, arguments);
        thisObj.render();
        if((typeof(Studio) != "undefined") && Studio)
        {
            /
            Register custom event or action here, and trigger the event afterwards.
             Studio.registerEvents(thisObj, "", "", EventConfig),
             Studio.registerAction(thisObj, "", "", ActionConfig, $.proxy(this.Cbk, this), );
             thisObj.triggerEvent("", )
             */
        }
    },

    /
    Triggered from init method and is used to render the widget
     */
    render : function()
    {
        var thisObj = this;
        var widgetProperties = thisObj.getProperties();
        var elem = thisObj.getContainer();
        var items = thisObj.getItems();
        var connectorProperties = thisObj.getConnectorProperties();

        /
        API to get base path of your uploaded widget API file
         */
        var widgetBasePath = thisObj.getWidgetBasePath();
        if(elem)
        {
```



```
            var vm = new Vue({
                el: $("#widget_demo_property", elem)[0],
                data: {
                    form: {
                        textProperty: widgetProperties.textProperty,
                        checkboxProperty: widgetProperties.checkboxProperty,
                        selectProperty: widgetProperties.selectProperty
                    }
                }
            })
        }

        /
        API to bind global events to the item DOM, it should not be deleted if there will some events to trigger in this widget.
         */
        thisObj.sksBindItemEvent();

        /
        API to refresh the previously bound events when a resize or orientation change occurs.
         */
        $(window).resize(function() {
            thisObj.sksRefreshEvents();
        });
    },
});
```

3. 在“widget_demo_property.ftl”中，定义渲染页面，属性配置为只读模式。

```
<div id="widget_demo_property">
    <el-form :model="form">
      <el-form-item label="Text Property">
        <el-input v-model="form.textProperty" :readonly="true"></el-input>
      </el-form-item>
     <el-form-item label="Checkbox Property">
        <el-input v-model="form.checkboxProperty" :readonly="true"></el-input>
      </el-form-item>
       <el-form-item label="Select Property">
        <el-input v-model="form.selectProperty" :readonly="true"></el-input>
      </el-form-item>
    </el-form>
</div>
```

4. 将开发好的组件代码压缩到后缀为“.zip”的压缩文件中，也可以单击**链接**，获取组件样例包“widget_demo_property.zip”。

步骤3 在组件库中上传组件包。

1. 在环境配置的“维护 > 全局元素 > 页面资产管理 > 组件”中，单击“提交新组件”。
2. 在提交新组件页面，设置组件基本信息，并上传压缩文件，单击“提交”。

图 8-3 上传自定义组件示例

表 8-1 上传自定义组件参数说明

参数	说明	示例
名字	新提交组件的名称，系统会根据组件包名称自动填充。	widgetdemoproperty
上传源文件	组件源文件包。	选择**步骤2**中的widget_demo_property.zip
场景	组件包的应用场景，可同时选择多个。	高级页面

参数	说明	示例
发行说明	组件的描述信息，按需进行设置。此处配置的信息，将会在组件详情页的“概况”页签中进行显示。	自定义组件

步骤4 关闭Vue3框架渲染组件开关。

本实践所涉及到的自定义组件是基于Vue2框架开发的，而系统是默认开启Vue3框架渲染组件的，所以您需要手动关掉Vue3框架渲染组件开关，否则拖拽组件到页面时会提示如下报错。

图 8-4 界面报错

1. 在应用设计器中，单击左侧导航栏中的“设置”。
2. 在“高级设置”中，取消勾选“页面组件的渲染框架由Vue2升级为Vue3”开关。

图 8-5 取消选中

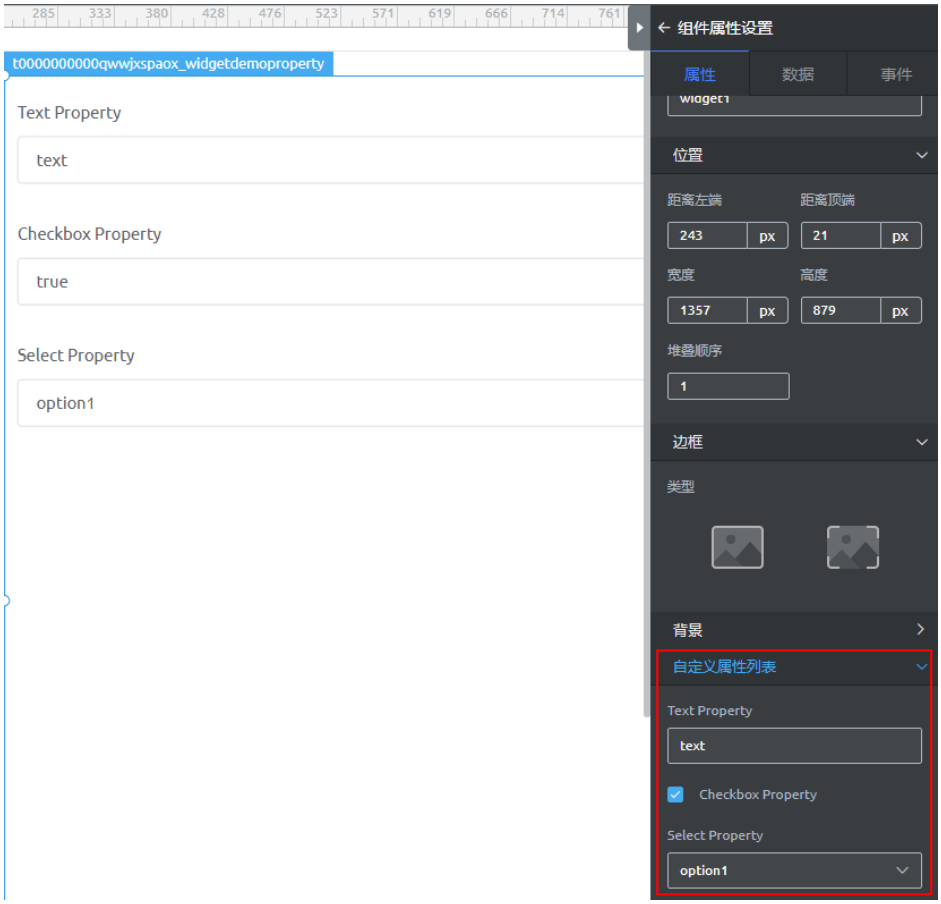
步骤5 在应用设计器中，选择“界面”，单击高级页面后的“+”，新建一个高级页面。

步骤6 在设计页面左上方单击  ，从“自定义”组件中拖拽**步骤3**中的组件至右侧空白页面。

步骤7 选中该组件，会在右侧显示该组件的属性配置面板。

步骤8 在“属性”页签下，“自定义属性列表”区域中可看到自定义的三个属性，根据需要进行修改。

图 8-6 编辑该 Widget 属性



----结束

8.2 使用 AstroZero 为组件配置中英文语言属性

期望实现效果

为组件配置多语言属性，实现组件在不同语种环境下都可正常显示。在AstroZero中对组件进行国际化配置，主要是修改国际化资源文件（i18n）。下面以为组件widget_demo_i18n配置中文和英文两种语言属性为例，其最终实现效果如图8-7、图8-8。

图 8-7 中文环境下显示效果

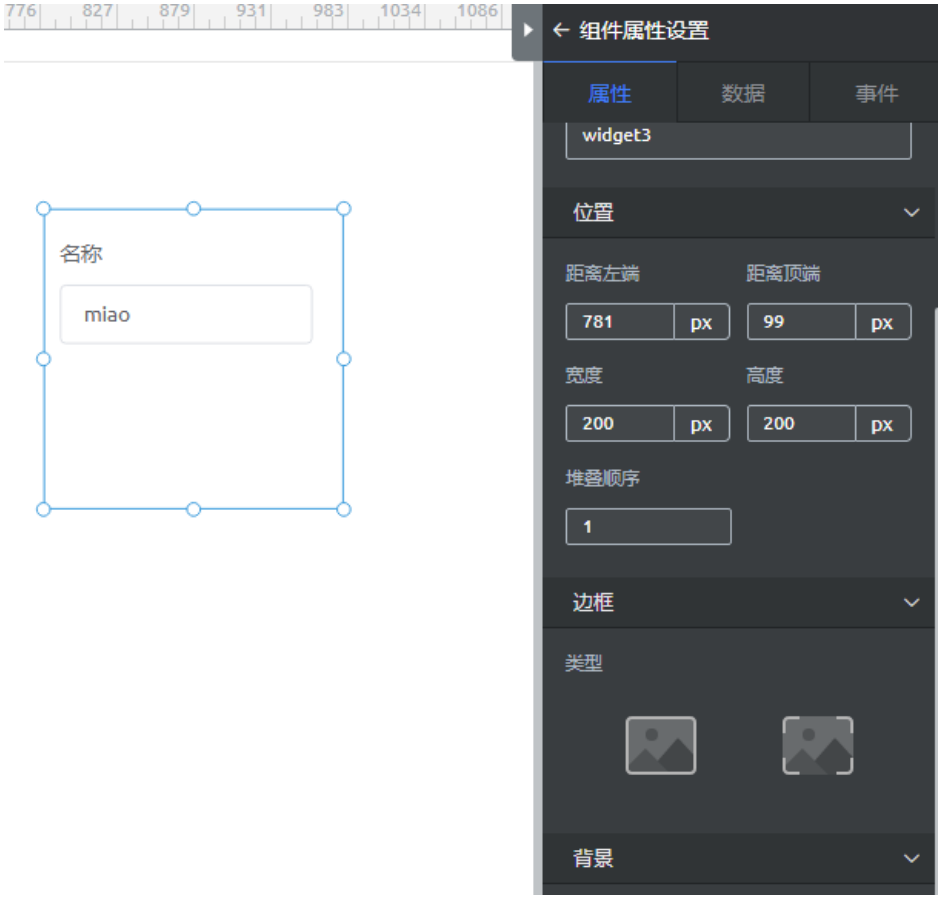
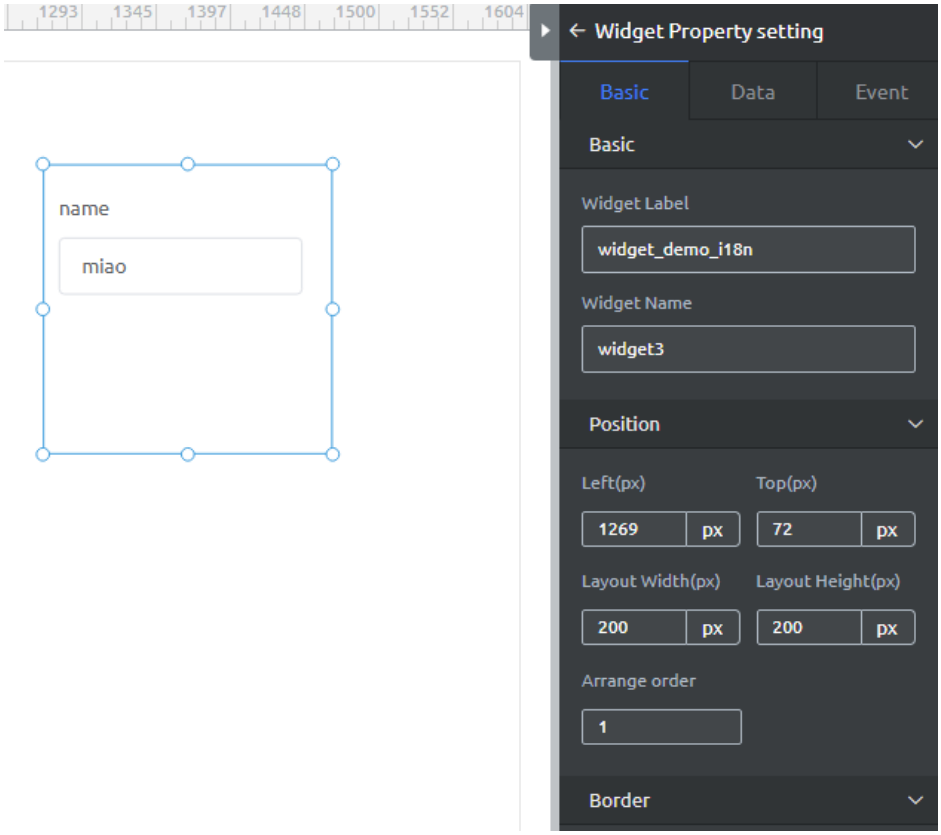


图 8-8 英文环境下显示效果



功能实现方法

步骤1 下载组件模板。

1. 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。
2. 单击，选择“环境管理 > 环境配置”，进入环境配置。
3. 在主菜单中，选择“维护”。
4. 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 组件模板”。
5. 在组件列表中，单击widgetVueTemplate，进入模板详情页。
6. 单击“下载”，设置组件的名称为“widget_demo_i18n”，单击“保存”，将模板下载到本地。

   如果选择“下载原始模板”，下载到本地的组件包中，组件名称不会被修改。

图 8-9 保存模板

步骤2 为组件设置多语言属性。

1. 增加“messages-en.json”和“messages-zh.json”国际化资源文件，“messages-zh.json”中需要使用unicode编码。
   - “messages-zh.json”文件内容如下：

     ```
     {
       "zh-CN": {
         "name" : "\u540d\u79f0"
       }
     }
     ```

   - “messages-en.json”文件内容如下：

     ```
     {
       "en-US": {
         "name" : "name"
       }
     }
     ```

2. 在“packageinfo.json”文件中，增加i18n节点，指定国际化资源文件的文件名。并增加requires节点，指定需要依赖的Vue和VueI18n库。

   其中，库文件名称和版本号，可在库详情页面获取。

   ```
   {
    "widgetApi": [
      {
        "name": "widget_demo_i18n"
      }
    ],
    "widgetDescription": "widget i18n demo",
    "authorName": "test",
    "width": "",
    "height": "",
    "i18n": [
      {
        "name": "messages-en"
      },
      {
        "name": "messages-zh"
      }
    ],
    "requires": [{
        "name": "global_Vue",
        "version": "100.7"
      },
      {
        "name": "global_VueI18n",
        "version": "100.7"
      },
      {
        "name": "global_Element",
   ```

```
    "version": "100.8"
  }
 ]
}
```

在“requires”里增加库文件时，需要注意某些库文件之间有依赖关系，增加库文件需要有先后顺序，例如“global_VueI18n”是基于“global_Vue”的，需要写在“global_Vue”之后。

3. 在“widget_demo_i18n.js”的render方法中，使用平台提供的“HttpUtils.getI18n”方法返回一个“i18n”变量，并新建Vue实例传入该“i18n”变量。

```
        var i18n = HttpUtils.getI18n({
            locale: HttpUtils.getLocale(),
            messages: thisObj.getMessages()
        });

        var vm = new Vue({
            el: $("#widget_demo_i18n", elem)[0],
            i18n: i18n,
            data: {
                form: {
                    name: "miao"
                }
            }
        })
```

4. 在“widget_demo_i18n.ftl”中，通过VueI18n提供的“$t”方法，使用国际化资源。

```
<div id="widget_demo_i18n">
    <el-form :model="form" :inline="true">
        <el-form-item :label="$t('name')">
            <el-input v-model="form.name"></el-input>
        </el-form-item>
    </el-form>
</div>
```

5. 将开发好的组件代码压缩到后缀为“.zip”的压缩文件中，也可以单击**下载链接**，获取样例包“widget_demo_i18n.zip”。

步骤3 在组件库中上传组件包。

1. 在环境配置的“维护 > 全局元素 > 页面资产管理 > 组件”中，单击“提交新组件”。
2. 在提交新组件页面，设置组件基本信息，并上传压缩文件，单击“提交”。

图 8-10 上传自定义组件示例

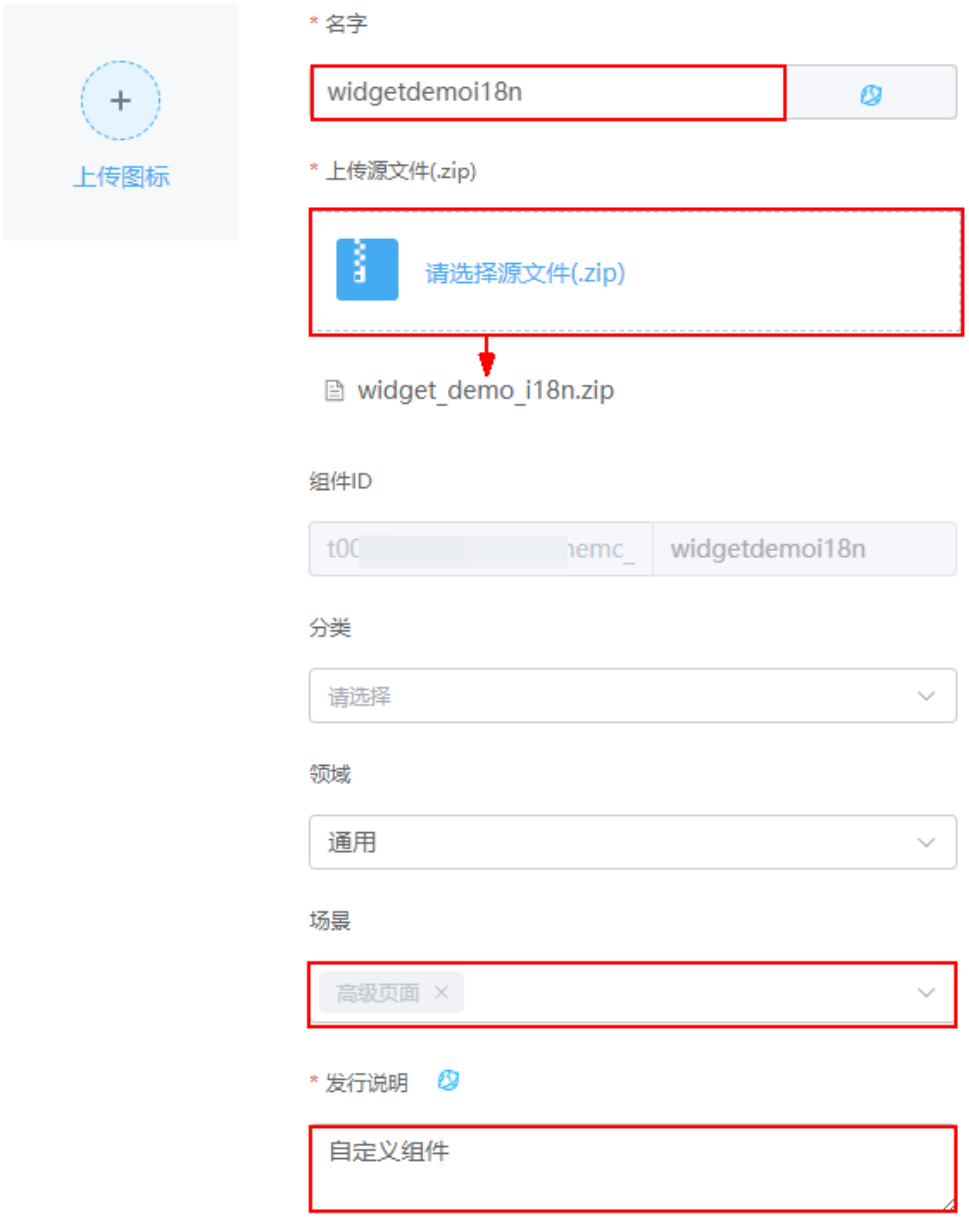


表 8-2 上传自定义组件参数说明

参数	说明	示例
名字	新提交组件的名称，系统会根据组件包名称自动填充。	widgetdemoi18n
上传源文件	组件源文件包。	选择**步骤2.5**中的widget_demo_i18n.zip
场景	组件包的应用场景，可同时选择多个。	高级页面

参数	说明	示例
发行说明	组件的描述信息，按需进行设置。此处配置的信息，将会在组件详情页的“概况”页签中进行显示。	自定义组件

步骤4 关闭Vue3框架渲染组件开关。

本实践所涉及到的自定义组件是基于Vue2框架开发的，而系统是默认开启Vue3框架渲染组件的，所以您需要手动关掉Vue3框架渲染组件开关，否则拖拽组件到页面时会提示如下报错。

图 8-11 界面报错

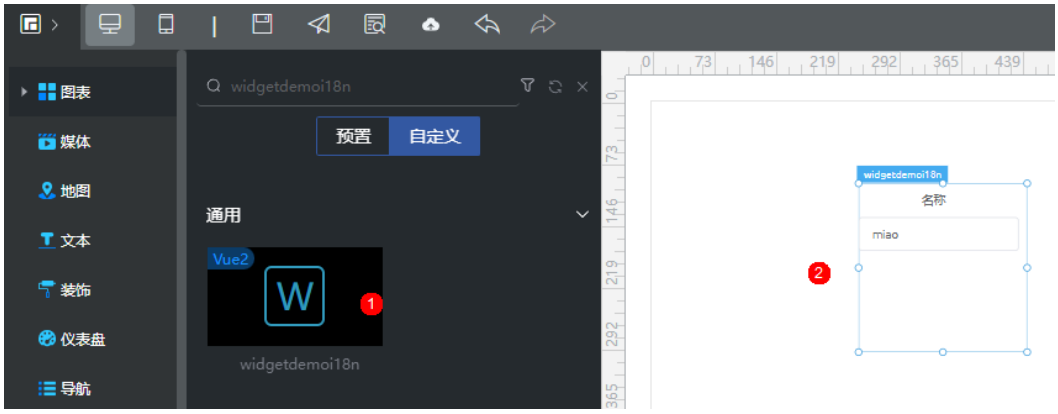
1. 在应用设计器中，单击左侧导航栏中的“设置”。
2. 在“高级设置”中，取消勾选“页面组件的渲染框架由Vue2升级为Vue3”开关。

图 8-12 取消选中

步骤5 在应用设计器中，选择“界面”，单击高级页面后的“+”，新建一个高级页面。

步骤6 单击  ，从“全部 > 自定义”中拖拽widgetdemoi18n组件到右侧画布中。

图 8-13 拖拽组件到画布



步骤7 单击，保存高级页面，保存成功后单击，发布高级页面。

步骤8 发布成功后，单击，预览效果。

切换环境语言，在不同语种下，在页面中选中Widget，会在右侧显示该组件的属性配置面板，查看组件属性是否符合预期。

----结束

8.3 使用 AstroZero 创建高级页面适配多终端显示

期望实现效果

当用户开发的高级页面应用于多种设备时，如何才能保证在不同大小的设备上，能够呈现同样的网页？为此，AstroZero提供了高级页面的电脑端和移动端两种终端视图、流式布局的响应式布局，并为绝对布局提供“拉伸”功能辅助自适应。另外，为确保自定义组件能够自适应不同分辨率设备，在组件开发中用户需要遵从响应式布局设计规范。组件的响应式设计，是高级页面适配多终端的重要前提。下面以开发一个满足响应式布局的商品列表组件为例，向您介绍如何适配多终端。

商品列表组件可以应用于不同分辨率的手机端和电脑端，组件中商品排布能够根据屏幕或者浏览器窗口大小自动调节。

功能实现方法

步骤1 下载组件模板。

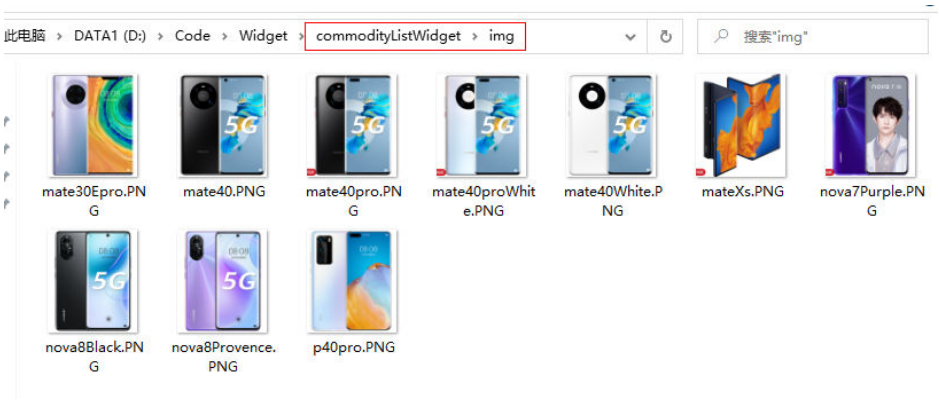
1. 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。
2. 单击，选择“环境管理 > 环境配置”，进入环境配置。
3. 在主菜单中，选择“维护”。
4. 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 组件模板”。
5. 在组件列表中，单击widgetVueTemplate，进入模板详情页。
6. 单击“下载”，设置组件的名称为“commodityListWidget”，单击“保存”，将模板下载到本地。

   如果选择“下载原始模板”，下载到本地的组件包中，组件名称不会被修改。

图 8-14 保存模板

步骤2 解压下载到本地的模板包，新建“img”文件夹，存放商品图片，此处存放了10张商品图片，分辨率统一为380x440，如图8-15所示。

图 8-15 新建 img 存在商品图片



步骤3 在commodityListWidget.js中，新建Commodity对象并为其赋值。

其中，Commodity对象包含三个属性，分别为src（商品图片路径）、title（商品标题）和content（商品描述）。

```
var vm = new Vue({
    el: $("#widgetVueTemplate",elem)[0],
    data:{
        Commodity: [
            {
                src: widgetBasePath+"img/mate40.PNG",
                title: "HUAWEI Mate 40",
                content: "HUAWEI Mate 40 5G 全网通 8GB+128GB（亮黑色）"
            },
            {
                src: widgetBasePath+"img/mate40White.PNG",
                title: "HUAWEI Mate 40",
                content: "HUAWEI Mate 40 5G 全网通 8GB+128GB（釉白色）"
            },
            {
                src: widgetBasePath+"img/mate40pro.PNG",
                title: "HUAWEI Mate 40 Pro",
                content: "HUAWEI Mate 40 Pro 全网通 8GB+128GB（亮黑色）"
            },
            {
                src: widgetBasePath+"img/mate40proWhite.PNG",
                title: "HUAWEI Mate 40 Pro",
                content: "HUAWEI Mate 40 Pro 全网通 8GB+256GB（秘银色）"
            },
```

```
        {
            src: widgetBasePath+"img/mateXs.PNG",
            title: "HUAWEI Mate Xs",
            content: "HUAWEI Mate Xs 5G 全网通 8GB+512GB（星际蓝）"
        },
        {
            src: widgetBasePath+"img/mate30Epro.PNG",
            title: "HUAWEI Mate 30E Pro",
            content: "HUAWEI Mate 30E Pro全网通 8GB+128GB（星河银）"
        },
        {
            src: widgetBasePath+"img/p40pro.PNG",
            title: "HUAWEI P40 Pro",
            content: "HUAWEI P40 Pro 5G 全网通 8GB+256GB（零度白）"
        },
        {
            src: widgetBasePath+"img/nova8Black.PNG",
            title: "HUAWEI nova 8",
            content: "HUAWEI nova 8 8GB+128GB 全网通版（亮黑色）"
        },
        {
            src: widgetBasePath+"img/nova8Provence.PNG",
            title: "HUAWEI nova 8",
            content: "HUAWEI nova 8 8GB+128GB 全网通版（普罗旺斯）"
        },
        {
            src: widgetBasePath+"img/nova7Purple.PNG",
            title: "HUAWEI nova 7",
            content: "HUAWEI nova 7 8GB+128GB 全网通版（仲夏紫）"
        }
    ]
  }
});
```

步骤4 在commodityListWidget.ftl中定义渲染页面，遍历Commodity对象，将多个商品的图片、标题和描述显示于组件中。

```
<div id="widgetVueTemplate" style="margin: 20px;">
  <section id="boxes">
    <div class="container">
      <div class="box" v-for="obj in Commodity">
        <img :src="obj.src">
        <h3>{{obj.title}}</h3>
        <p>{{obj.content}}</p>
      </div>
    </div>
  </section>
</div>
```

步骤5 在commodityListWidget.css中，定义组件的样式。

其中，采用百分比调节单个商品宽度，并设置断点，通过@media查询改变不同视区宽度下商品显示宽度的百分比。

```
/*设置组件字体、内边距、外边距和背景颜色*/
#widgetVueTemplate{
  font:15px/1.5 Arial, Helvetica, 'Microsoft YaHei';
  padding:0;
  margin:0;
  background-color: #f4f4f4;
}

/*商品列表*/
#boxes{
  margin-top:10px;
}

/*商品列表宽度为boxes宽度80%*/
.container{
  width:80%;
  margin: auto;
```



```
    overflow: hidden;
}
/*单商品样式 默认宽度为.container宽度18%，单行商品最大个数：5*/
#boxes .box{
    background: #ffffff;
    float:left;
    text-align: center;
    width:18%;
    padding:5px;
    margin: 7px 7px 7px 7px;
}

/*商品图片样式 默认宽度为.box宽度95%，为保证在屏幕宽度较大时，图片比例合理，限制其最大宽度为200px*/
#boxes .box img{
    width: 95%;
    max-width:200px;
    margin-top: 5px;
}

/*商品描述样式 防止描述文字行数不同导致的商品高度不同而产生排列错位，限制其最小高度为44px*/
#boxes .box p{
    min-height: 44px;
}

/*Media Queries 根据max-width（视区最大宽度）设置断点，调节单个商品的宽度以改变单行商品最大个数*/
/*视区宽度介于1600px-1700px，单行商品最大个数：4*/
@media(max-width:1700px){
    #boxes .box{
        float:left;
        text-align: center;
        width: 22%;
    }
}
/*视区宽度介于1400px-1600px，单行商品最大个数：3*/
@media(max-width:1600px){
    #boxes .box{
        float:left;
        text-align: center;
        width: 30%;
    }
}
/*视区宽度介于800px-1400px，单行商品最大个数：2*/
@media(max-width:1400px){
    #boxes .box{
        float:left;
        text-align: center;
        width: 44%;
    }
}
/*视区宽度800px以下，单行商品最大个数：1*/
@media(max-width:800px){
    #boxes .box{
        float:none;
        text-align: center;
        width: 95%;
    }
}
```

步骤6 设置完成后，重新打包（包名为commodityListWidget.zip）。

您也可以直接单击**commodityListWidget.zip**，下载commodityListWidget商品列表组件示例开发包。

步骤7 打包上传commodityListWidget商品列表组件。

1. 在环境配置页面的主菜单中，选择“维护”。
2. 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 组件”。

3. 单击“提交新组件”，进入提交新组件页面。
4. 单击“请选择源文件(.zip)”，选择步骤6中的商品列表组件包commodityListWidget.zip，输入发行说明为“commodityListWidget”，单击“提交”。

步骤8 关闭Vue3框架渲染组件开关。

本实践所涉及到的自定义组件是基于Vue2框架开发的，而系统是默认开启Vue3框架渲染组件的，所以您需要手动关掉Vue3框架渲染组件开关，否则拖拽组件到页面时会提示如下报错。

图 8-16 界面报错

1. 在应用设计器中，单击左侧导航栏中的“设置”。
2. 在“高级设置”中，取消勾选“页面组件的渲染框架由Vue2升级为Vue3”开关。

图 8-17 取消选中

步骤9 创建CommodityDisplay高级页面。

首次创建高级页面时，会出现视图选项，请勾选“电脑端”和“手机端”，布局类型选择“流式布局”（流式布局为自适应布局）。

图 8-18 新建 CommodityDisplay 高级页面

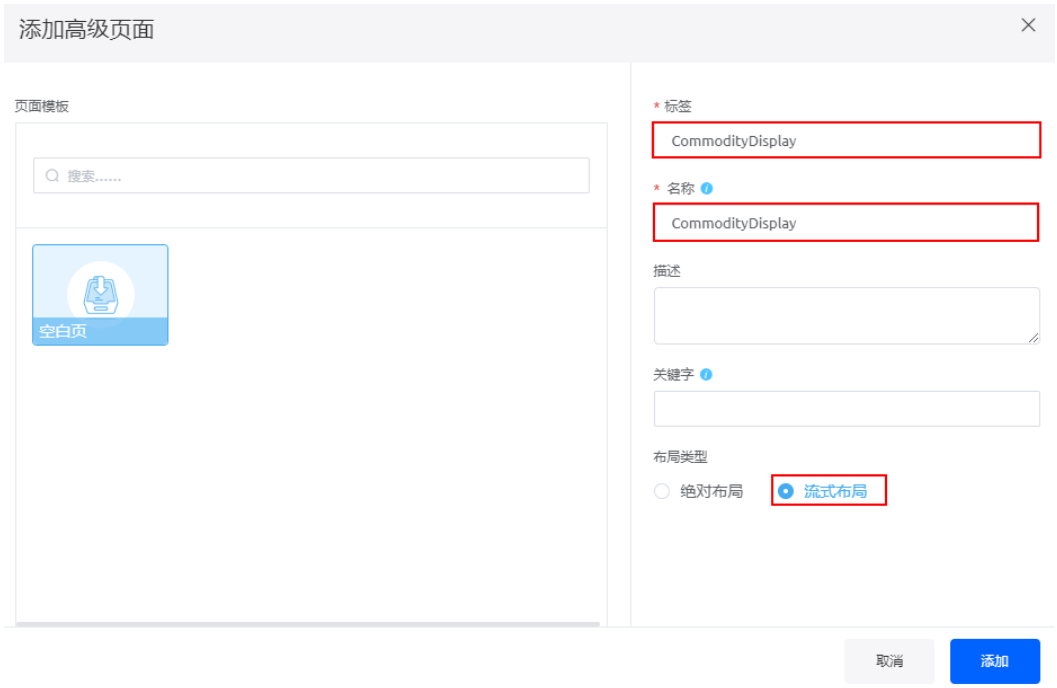


表 8-3 创建高级页面 CommodityDisplay 参数说明

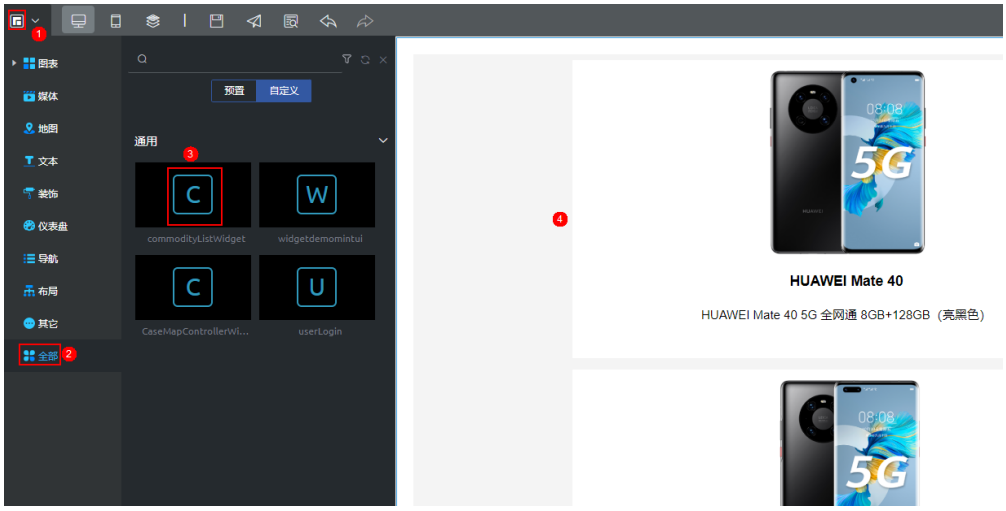
参数	说明	示例
标签	新建高级页面的标签，创建后可修改。 取值范围：1~100个字符。	CommodityDisplay
名称	新建高级页面的名称，创建后不可修改。命名要求如下： • 1~100个字符。 • 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	CommodityDisplay

参数	说明	示例
布局类型	选择高级页面的布局模式。 • 绝对布局：在绝对布局中，每个组件可在页面中任意位置进行拖拽放置，组件的宽高可自定义设置。 • 流式布局：在流式布局中，拖拽到页面中的组件，将根据从上到下、从左到右的顺序依次排列，组件的高度将根据组件内容大小进行自适应，宽度可按百分比进行配置。	流式布局

步骤10 开发CommodityDisplay电脑端和移动端高级页面。

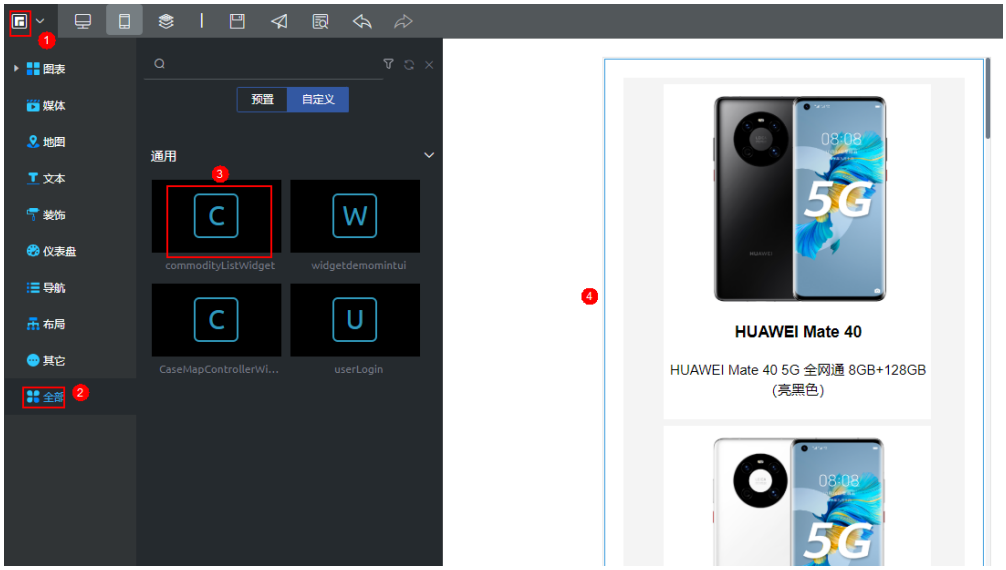
1. 在设计页面左上方，单击，从“全部 > 自定义”中拖拽commodityDetailsWidget组件到右侧画布中。

图 8-19 在电脑端拖拽 CommodityDisplay 组件



2. 在页面上方单击，切换到移动端，并拖拽commodityListWidget组件到右侧画布中。

图 8-20 在移动端拖拽 CommodityDisplay 组件



3. 单击页面上方的，保存高级页面后，单击，发布高级页面。
4. 单击，进入预览页面，调整窗口大小，查看效果是否符合预期。

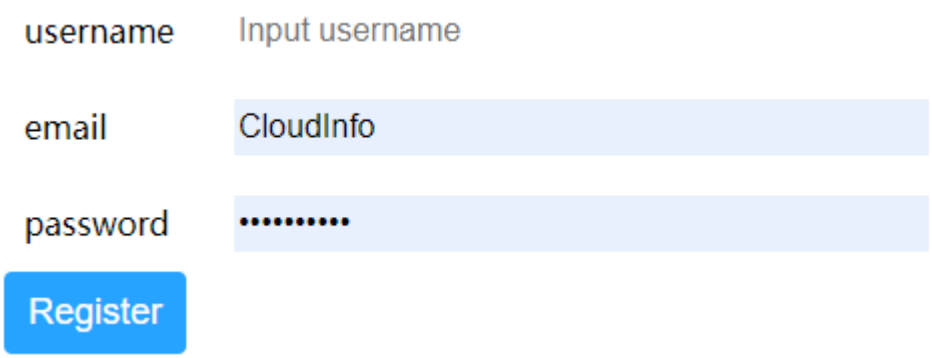
----结束

8.4 使用 AstroZero 开发高级页面时如何引用第三方库

期望实现效果

在自定义高级页面组件开发过程中，AstroZero支持开发者直接引用第三方库，在降低组件开发复杂度的同时，丰富了组件的功能。库是支撑高级页面组件运行的第三方依赖，如果缺少相应的库，则高级页面组件不能正常运行。AstroZero提供了一些系统预置库，可在高级页面组件中直接引用或在页面设置中直接进行加载并使用。当系统预置的库无法满足需求时，可以上传自定义库，并加载到页面中使用。以某组件中需要使用Vue（系统预置库）以及MintUI（自定义库）为例，介绍如何上传自定义库，并在组件中使用库，最终实现效果如图8-21所示。

图 8-21 引入库后效果

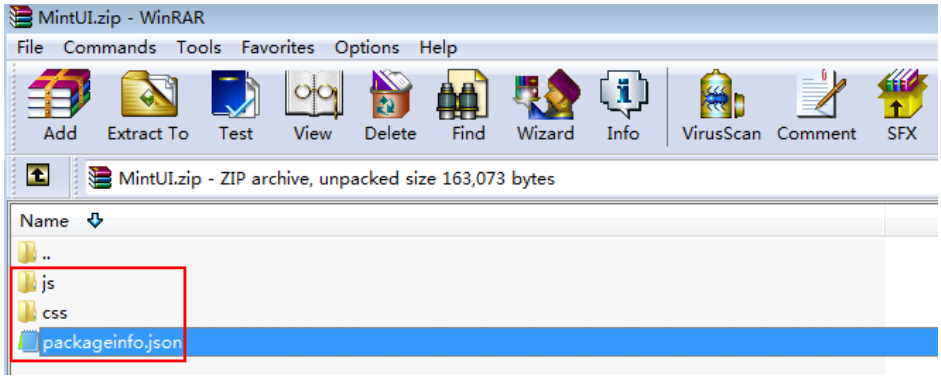


功能实现方法

步骤1 将自定义库的相关文件打成Zip包。

例如，在MintUI官网下载组件库的代码，增加一个packageinfo.json元数据描述文件，在该文件中列出Library包含的js和css文件名，并打成Zip包，如图8-22。您也可以单击链接，获取该包。

图 8-22 MintUI 库文件结构



packageinfo.json中必选要包含待引入的文件。例如，MintUI库需要引入文件“js/index.js”和“css/index.css”，请在packageinfo.json中添加这两个文件的描述。其中，“js”和“css”用于定义文件类型，“name”用于定义文件路径及名称。

```
{
  "js": [
    {
      "name": "js/index"
    }
  ],
  "css": [
    {
      "name": "css/index"
    }
  ]
}
```

步骤2 上传自定义库。

1. 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。
2. 单击，选择“环境管理 > 环境配置”，进入环境配置。
3. 在主菜单中，选择“维护”。
4. 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 库”。
5. 单击“提交新库”，进入提交新库页面。
6. 设置库的基本信息，上传步骤1中的zip包后，单击“提交”。

图 8-23 上传库

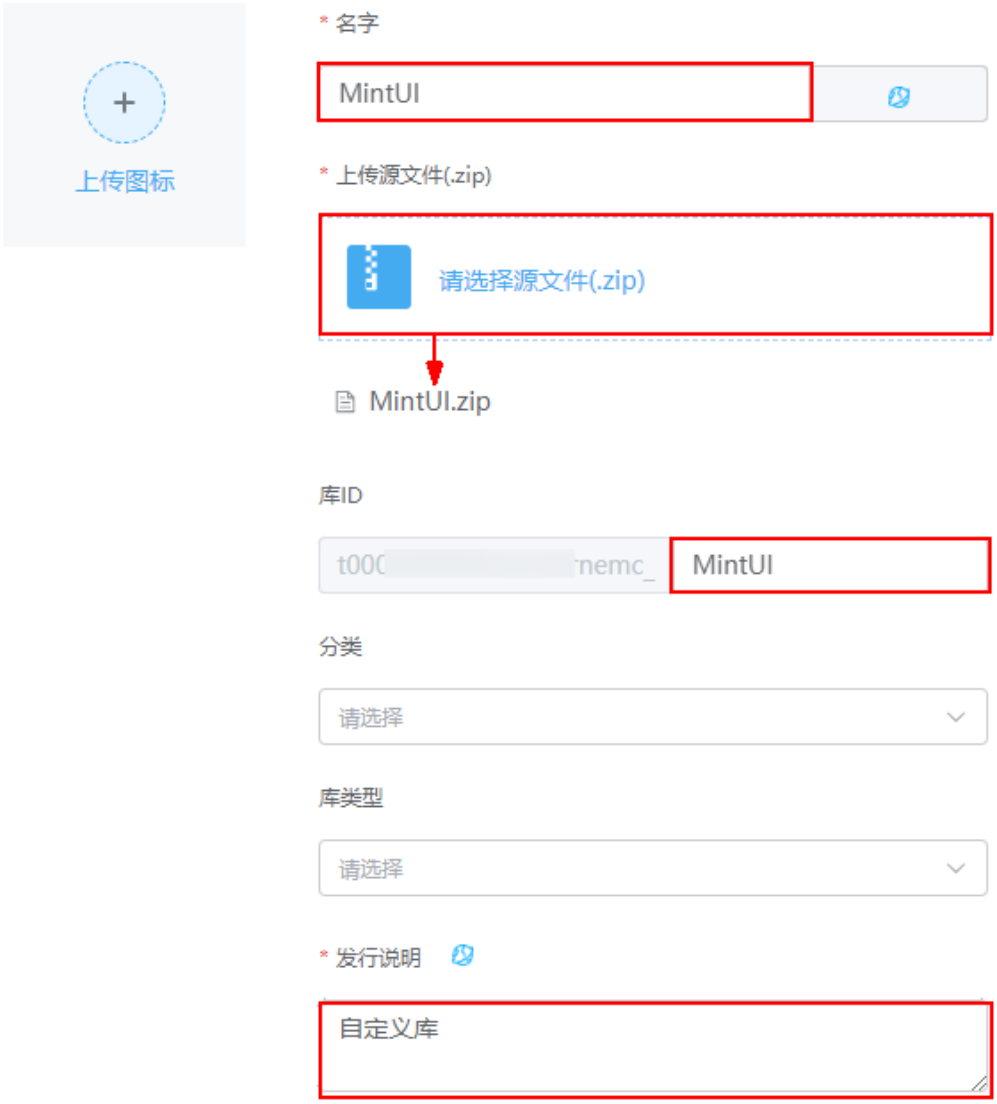


表 8-4 新建 MintUI 库参数说明

参数	说明	示例
名字	新建库的名称。 取值范围：1~80个字符。	MintUI
上传源文件	选择自定义库的压缩包。	选择**步骤1**中的库
库ID	上传库的ID，由字母及数字组成，且必须以字母开头。	MintUI
发行说明	自定义库的描述信息，按需进行设置。	自定义库

步骤3 在高级页面组件中，引用库。

以自定义组件（widget_demo_mintui）为例，在该组件引用第三方库。

1. 在组件包的“packageinfo.json”文件中，增加requires节点，指定需要依赖库的库ID和版本号。

   其中，“name”为库ID、“version”为库版本号数字部分。

   例如，增加如下requires节点，库文件名称和版本号在库详情页面获取。

```
"requires": [
  {
    "name": "global_Vue",
    "version": "100.7"
  },
  {
    "name": "t0000000000fcsfrfcaks_MintUI",
    "version": "1.0.0"
  }
 ]
```

   在“requires”里增加库文件时，需要注意某些库文件之间有依赖关系，增加库文件需要有先后顺序，例如“global_VueI18n”是基于“global_Vue”的，需要写在“global_Vue”之后。

2. 在高级页面组件包的widget_demo_mintui.ftl中，写一个简单的表单DOM。

```
<div id="widget_demo_mintui">
 <mt-field label="username" placeholder="Input username" v-model="username"></mt-field>
 <mt-field label="email" placeholder="Input email" type="email" v-model="email"></mt-field>
 <mt-field label="password" placeholder="Input password" type="password" v-
modal="password"></mt-field>
 <mt-button type="primary" @click="submit">Register</mt-button>
</div>
```

3. 在高级页面组件包的widget_demo_mintui.js/render方法中，新增Vue实例。

```
Vue.use(MINT);
var vm = new Vue({
   el: $("#widget_demo_mintui", elem)[0],
   data:{
     username: "",
     email: "",
     password: ""
   },
   methods:{
     submit: function(){
       console.log(this.username + " registers");
     }
   }
})
```

4. 将修改后的内容，重新打包，您也可以单击**widget_demo_mintui.zip**，获取该组件包。

步骤4 返回环境配置，选择“维护 > 全局元素 > 页面资产管理 > 组件”，单击“提交新组件”，将自定义组件包上传至组件库。

图 8-24 上传组件

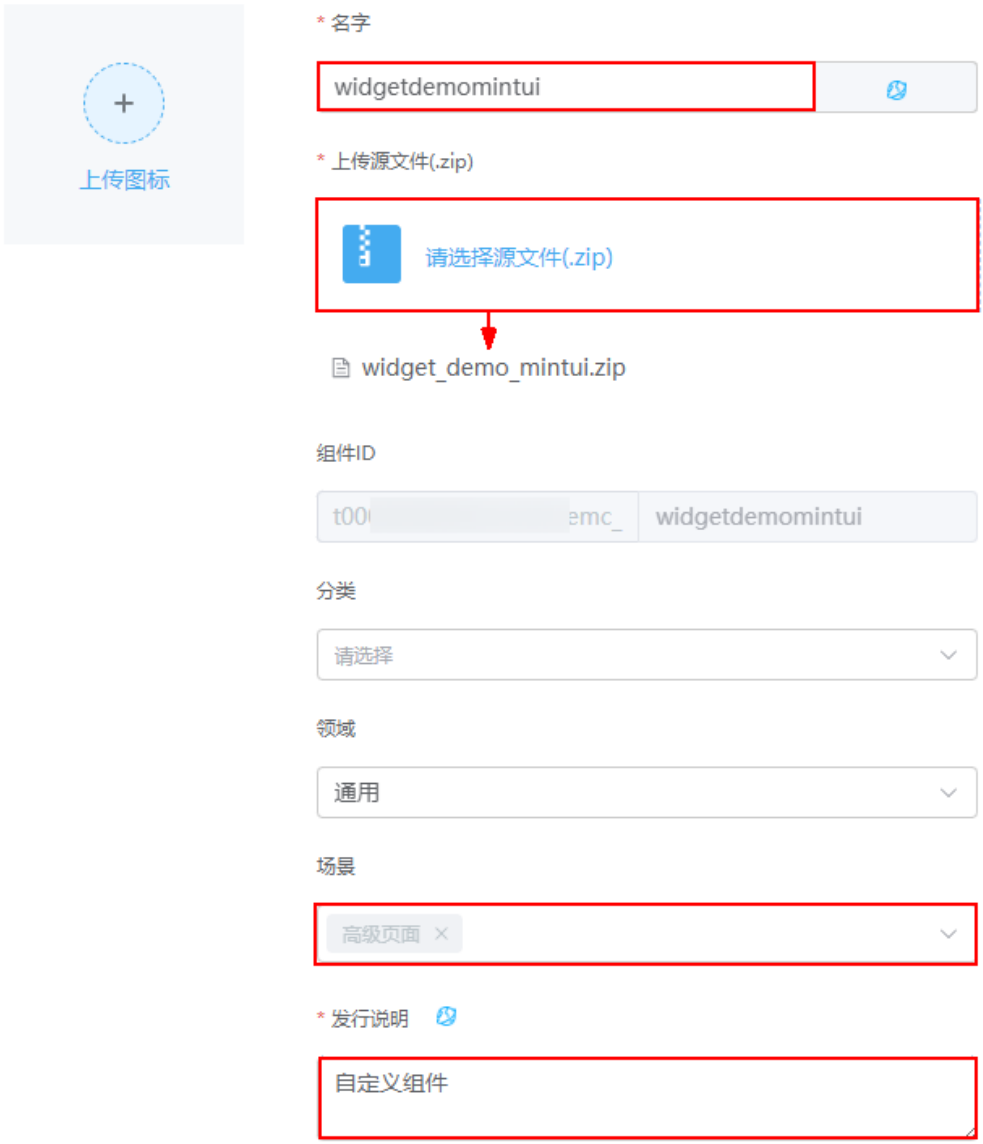


表 8-5 上传自定义组件参数说明

参数	说明	示例
名字	新提交组件的名称，系统会根据组件包名称自动填充。	widgetdemomintui
上传源文件	组件源文件包。	选择**步骤3.4**中的widget_demo_mintui.zip
场景	组件包的应用场景，可同时选择多个。	高级页面

参数	说明	示例
发行说明	组件的描述信息，按需进行设置。此处配置的信息，将会在组件详情页的“概况”页签中进行显示。	自定义组件

步骤5 关闭Vue3框架渲染组件开关。

本实践所涉及到的自定义组件是基于Vue2框架开发的，而系统是默认开启Vue3框架渲染组件的，所以您需要手动关掉Vue3框架渲染组件开关，否则拖拽组件到页面时会提示如下报错。

图 8-25 界面报错

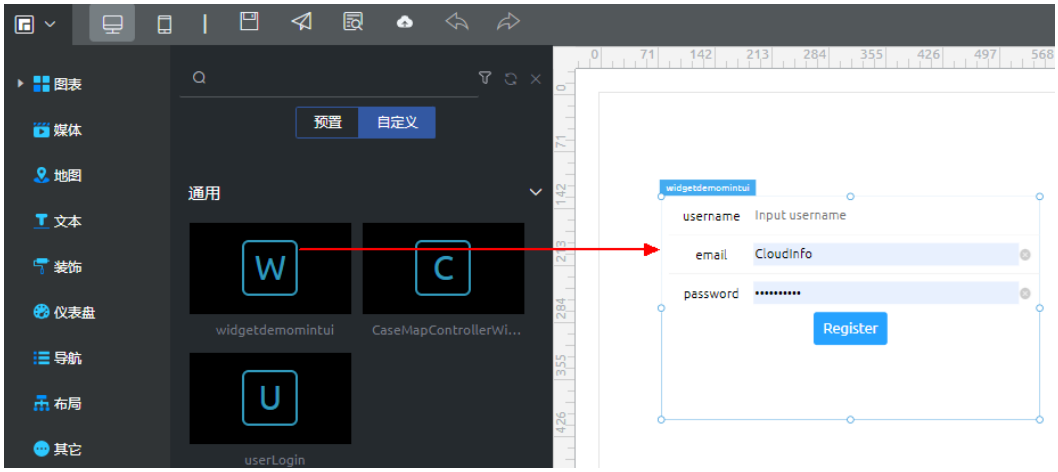
1. 在应用设计器中，单击左侧导航栏中的“设置”。
2. 在“高级设置”中，取消勾选“页面组件的渲染框架由Vue2升级为Vue3”开关。

图 8-26 取消选中

步骤6 在应用设计器中，选择“界面”，单击高级页面后的“+”，新建一个高级页面。

步骤7 单击  ，从“全部 > 自定义”中拖拽widget_demo_mintui组件到右侧画布中。

图 8-27 拖拽组件到画布中



步骤8 单击页面上方的，保存高级页面后，单击，发布高级页面。

步骤9 单击，进入预览页面，查看效果是否符合预期。

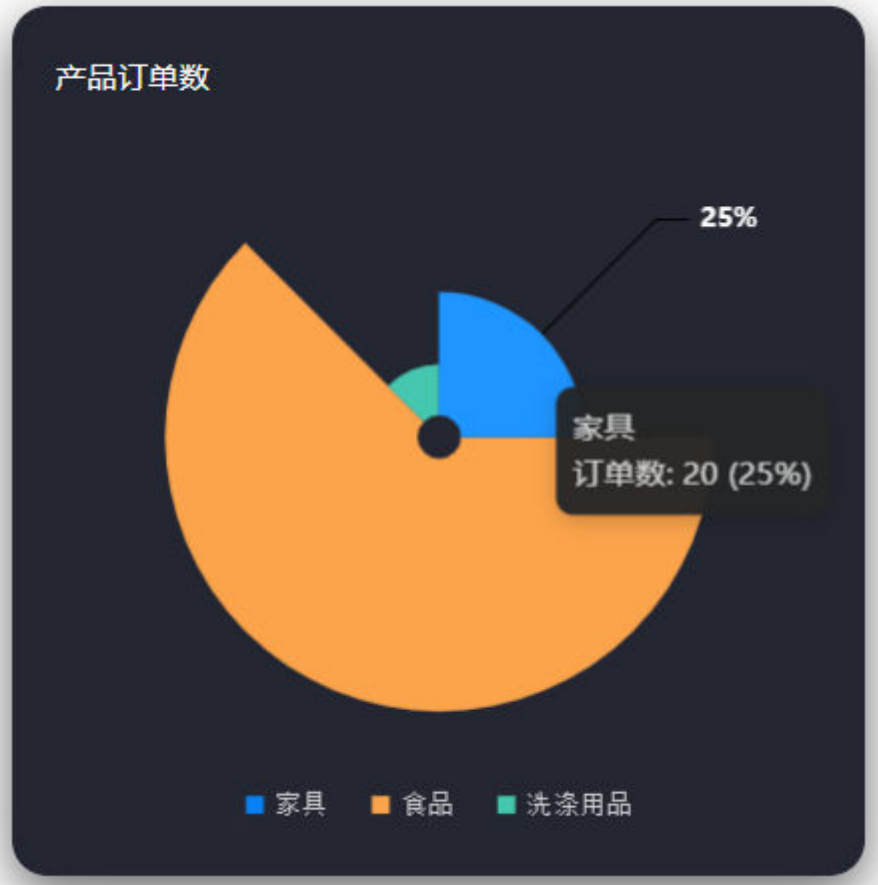
----结束

8.5 在 AstroZero 高级页面中使用花瓣图展示订单数据

期望实现效果

高级页面中组件展示的数据除了系统预置的静态数据之外，还支持动态数据，即通过调用脚本、服务编排或对象等接口动态生成的数据。例如，将玫瑰花饼图的数据修改为订单对象中的数据。

图 8-28 实现效果



功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 8-29 创建一个空白应用

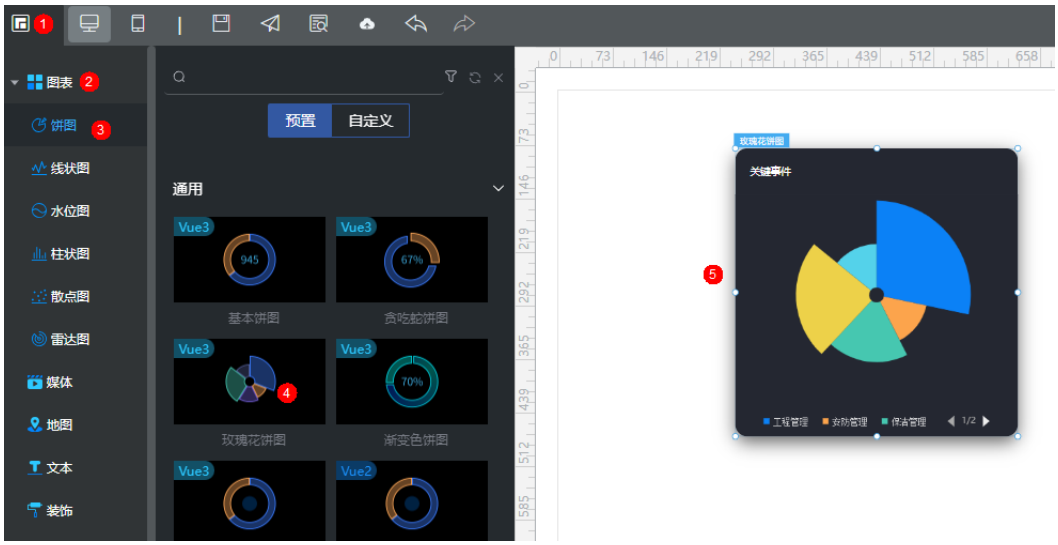
表 8-6 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 在应用设计器中，选择“界面”，单击高级页面后的“+”，新建一个高级页面。

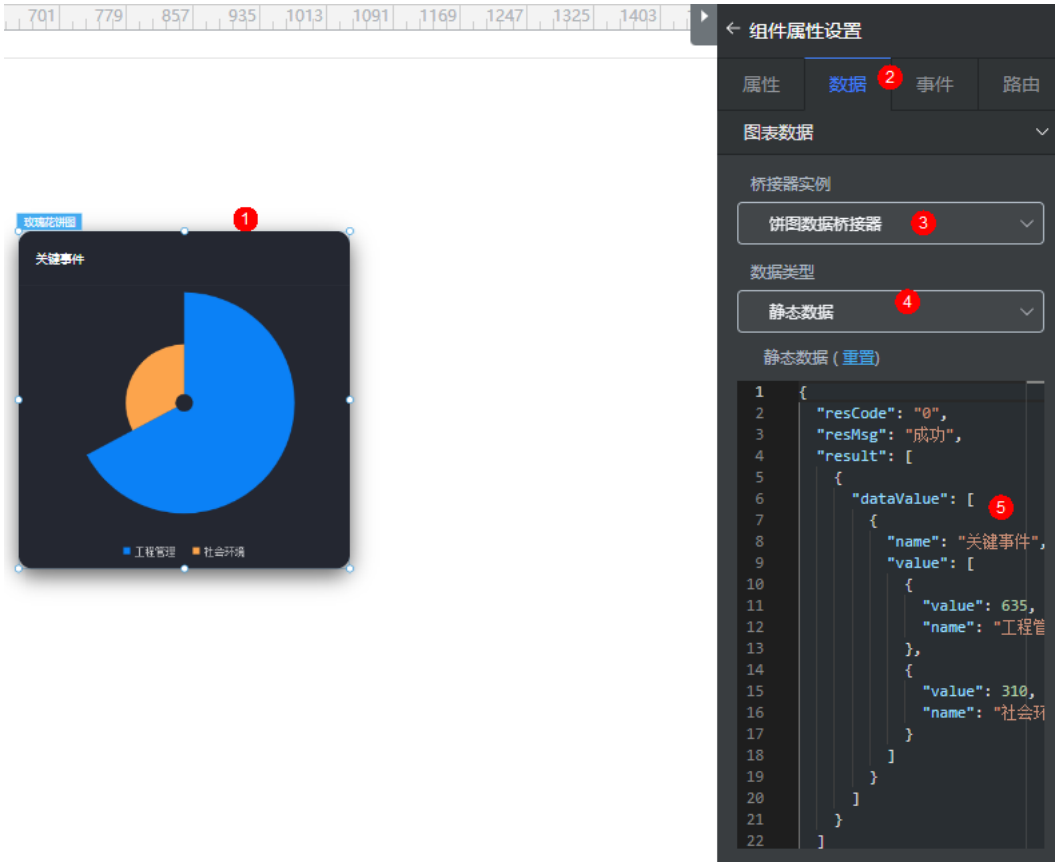
步骤3 单击，从“图表 > 饼图”中拖拽widget_demo_mintui组件到右侧画布中。

图 8-30 拖入一个玫瑰花饼图



步骤4 选中玫瑰花饼图，在“数据”中查看该组件对应的数据格式。

图 8-31 查看数据格式



步骤5 新建一个productList对象，并为对象添加字段和数据。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“productList”，单击“确定”。

图 8-32 创建对象 productList

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加productName字段。

图 8-33 添加 productName 字段

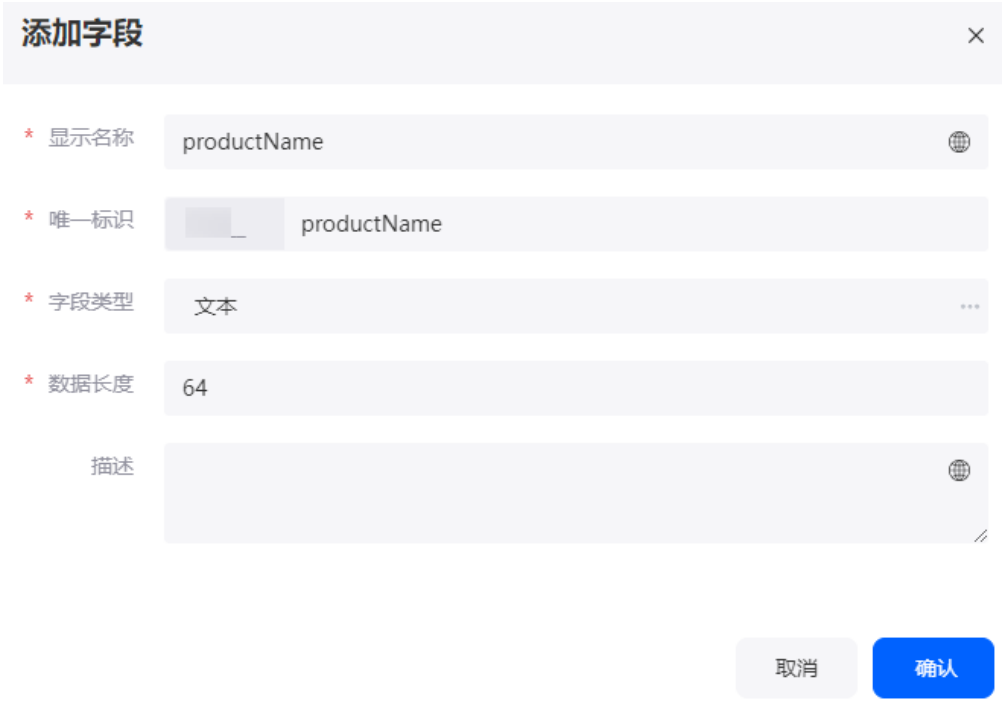


表 8-7 添加 productName 字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	productName

参数	说明	示例
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	productName
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

5. 在“字段”页签，再次单击“添加”按钮，添加productNumber字段。

图 8-34 添加 productNumber 字段

6. 在“数据”页签，单击“添加”，为对象添加数据。

图 8-35 为对象添加数据

步骤6 创建读取对象数据的脚本。

1. 在应用设计器的左侧导航栏中，选择“逻辑”，单击脚本中的“+”。
2. 创建一个空白脚本，名称为getDataInfo。

图 8-36 创建脚本 getDataInfo

3. 在脚本编辑器中，输入如下代码。

```
// Here's your code.
import * as db from 'db';
@useObject(['命名空间__productList__CST'])
@action.object({ type: 'method' })
export class SearchScript {
    @action.method({ input: 'ParamsInput', output: 'ParamsOutput' })
    public run(): Object[] {
        let queryData = this.doSearchScript();
        let result: Array<Object> = [];
        result.push({
            dataValue: [{
                name: "订单数",
                value: queryData
            }]
        });
        console.log("result", result)
        return result;
    }
    private doSearchScript(): Object[] {
        let sql = "select 命名空间__productName__CST as name,命名空间__productNumber__CST as
value"
```

```
            + " from 命名空间__productList__CST"
        let query = db.sql().exec(sql)
        return query;
    }
}
```

4. 单击，保存脚本，保存成功后单击，激活脚本。

步骤7 新建开放接口。

1. 在应用设计器中，选择“集成”。
2. 单击开放接口后的“+”，设置开放接口参数。

图 8-37 设置开放接口参数

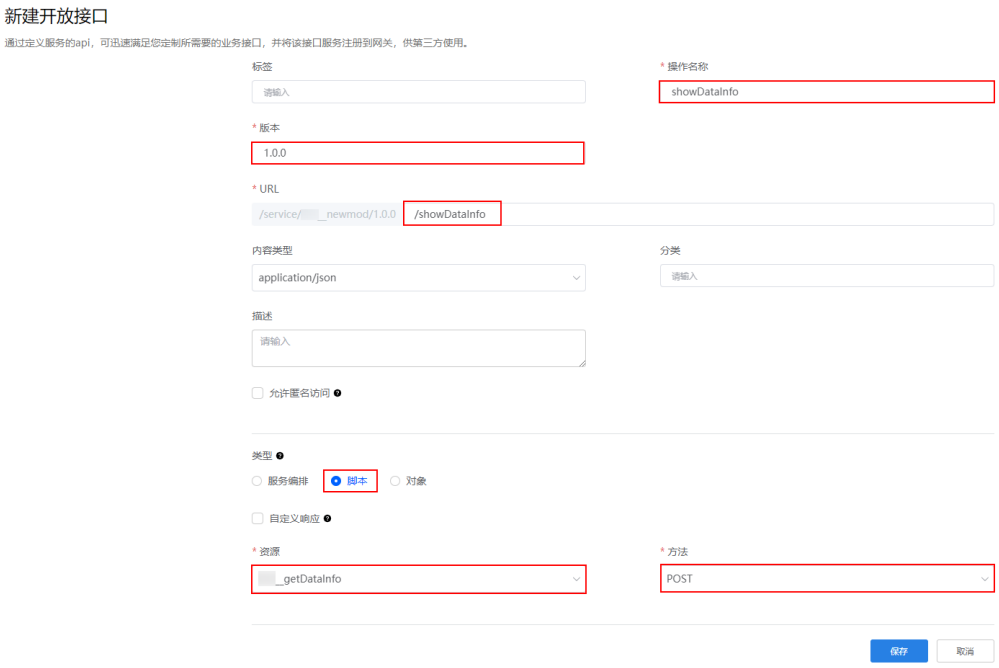


表 8-8 新建接口参数说明

参数	说明	示例
操作名称	新建接口的操作名称。 命名要求如下： – 取值范围：1~40个字符。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	showDataInfo
版本	URL对应的版本号。	1.0.0

参数	说明	示例
URL	新URL地址，其中“/service”是固定值，其次是“/*App名称*/版本号”，剩下部分请自定义。	/showDataInfo
类型	选择新建接口的类型。	脚本
资源	选择接口调用的脚本。	选择**步骤6**中创建的脚本
方法	映射后调用的方法名。	POST

3. 设置完成后，单击“保存”，进入接口详情页。
4. 在接口信息中，单击URL后的，复制URL。

图 8-38 复制 URL

步骤8 返回高级页面，选中“玫瑰花饼图”，在“数据”页签，设置数据桥接器。

“桥接器实例”选择“饼图数据桥接器”，“数据类型”选择“动态数据”，“URL”设置为**步骤7.4**中获取的URL。

图 8-39 设置组件数据

步骤9 在玫瑰花饼图组件上，单击右键选择“高级设置”，设置组件标题，字体大小等。

图 8-40 设置组件标题

步骤10 单击，保存高级页面，保存成功后单击，发布高级页面。

步骤11 发布成功后，单击，预览效果。

----结束

8.6 在 AstroZero 高级页面中使用轮播组件实现图片展示和 URL 跳转

期望实现效果

高级页面中的轮播组件主要用于多个图片的自动循环切换。您也可以为图片添加超链接，即单击图片，跳转到指定的网站。

功能实现方法

步骤1 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 8-41 创建一个空白应用

表 8-9 新建空白应用参数说明

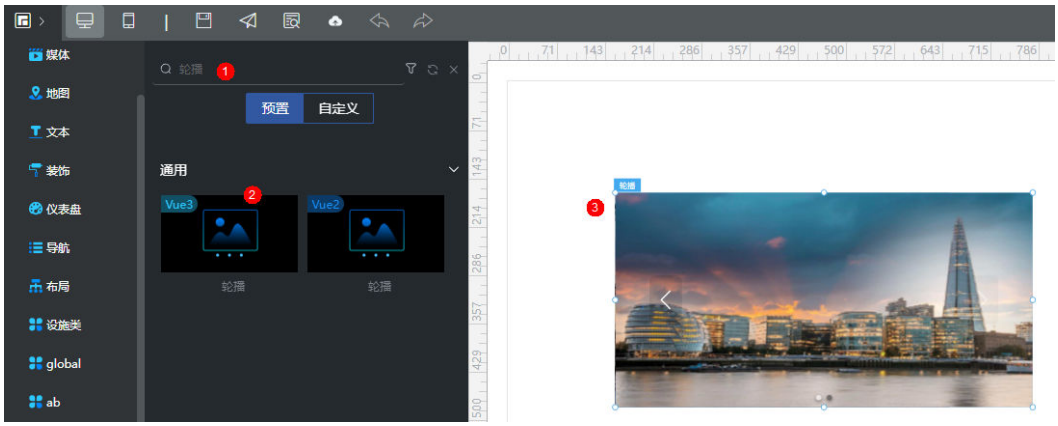
参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间__***。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤2 在应用设计器中，选择“界面”，单击高级页面后的“+”，新建一个高级页面。

步骤3 单击  ，拖拽轮播组件到右侧画布中。

图 8-42 拖入轮播组件



步骤4 在轮播组件上，单击右键选择“高级设置”。

步骤5 单击“轮播设置”，根据个人实际需求，设置轮播方向、图片播放时间间隔等。

图 8-43 选择轮播设置

图 8-44 轮播设置

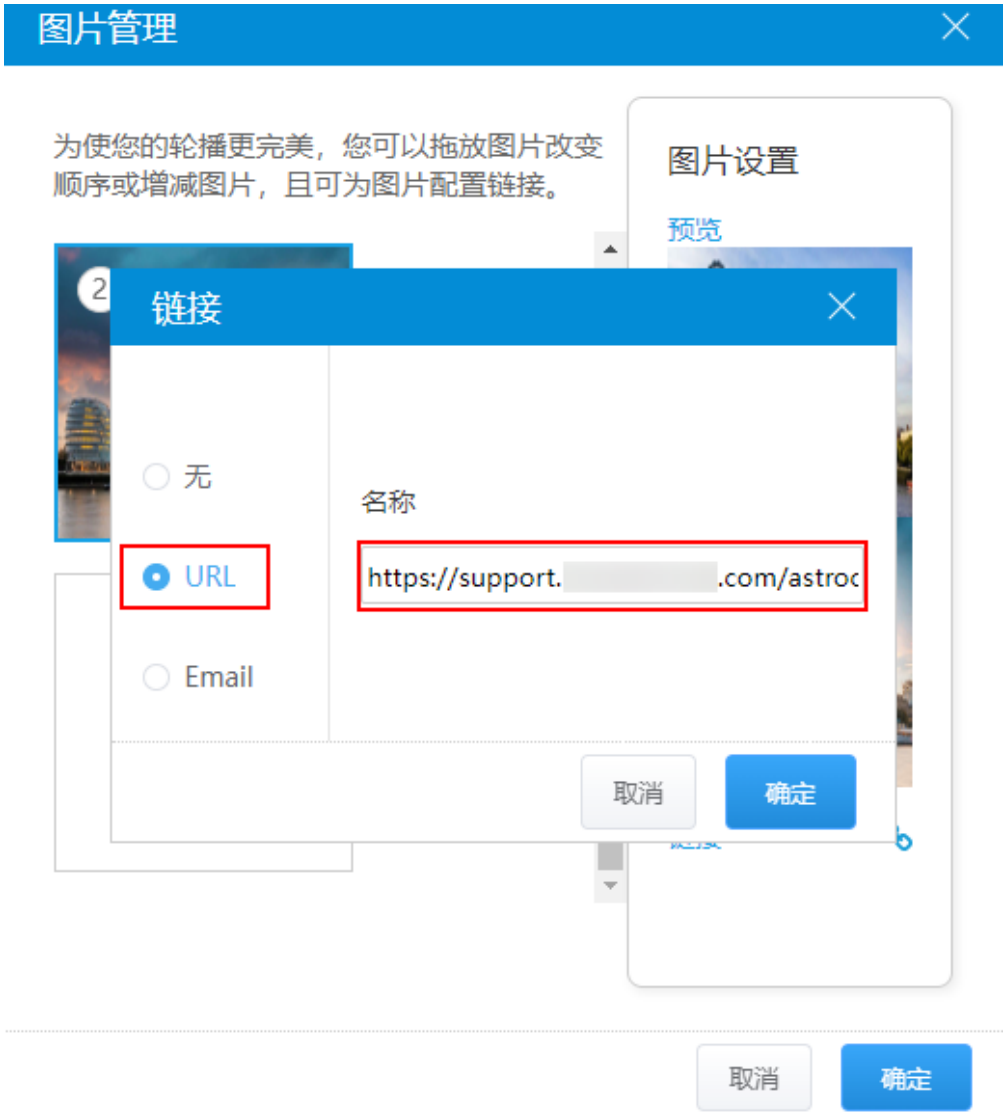
步骤6 新增轮播图片并设置URL跳转。

1. 再次选中轮播组件，单击右键选择“高级设置”，选择“图片管理”。

图 8-45 选择图片管理

2. 单击“添加”，选中所需的图片，单击“保存”。
3. 选中已添加的图片，在右侧图片设置中，单击链接后的 。
4. 设置跳转的URL地址，单击“确定”。

图 8-46 设置链接



步骤7 单击“确定”，返回高级页面。

步骤8 单击，保存高级页面，保存成功后单击，发布高级页面。

步骤9 发布成功后，单击，预览效果。

----结束

9 连接器专项

9.1 通过 AstroZero 中的连接器上传并识别身份证图片

期望实现效果

AstroZero封装了不同类型的连接器用于对接其他服务，对接后即可在应用中使用该服务。例如，通过连接器对接OCR后，可识别某用户上传到华为OBS上的身份证图片的全部信息。

图 9-1 在页面提交身份证信息

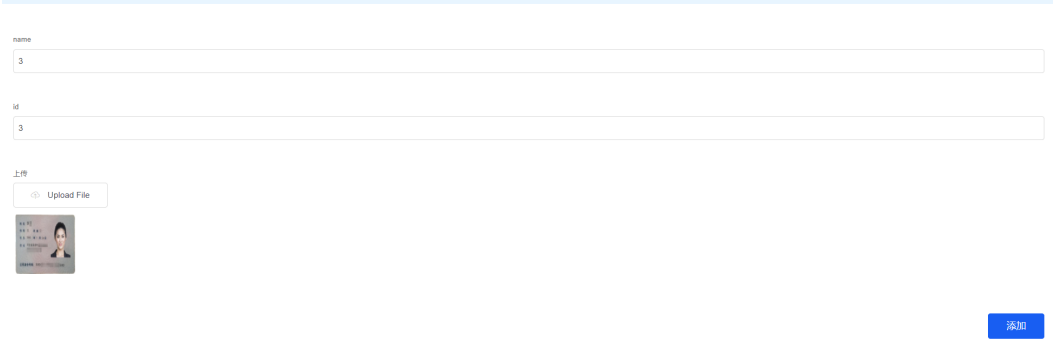


图 9-2 成功识别图片

功能实现方法

步骤1 准备工作。

- 需要拥有一个华为账号或一个可用于访问OBS的IAM用户，即先注册华为云并实名认证、创建IAM用户、充值以及购买资源包，具体操作请参见**使用OBS前需要做的准备工作**。
- 获取AK（Access Key ID）、SK（Secret Access Key），即访问密钥对，具体操作请参见**获取AK/SK**。
- 已在华为OBS上，创建存储桶（例如“appcubecn4”），用于后续存储对象使用，具体操作请参见**如何创建桶**，请记录创建桶时选择的区域。

步骤2 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 9-3 创建一个空白应用

表 9-1 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤3 创建一个OBS连接器。

1. 在应用设计器中，选择“集成”，单击“连接器”下的“连接器实例”。
2. 在存储中，选择“OBS”，单击“+”，进入创建OBS页面。
3. 输入基本信息并添加桶，单击“保存”。

图 9-4 设置基本信息

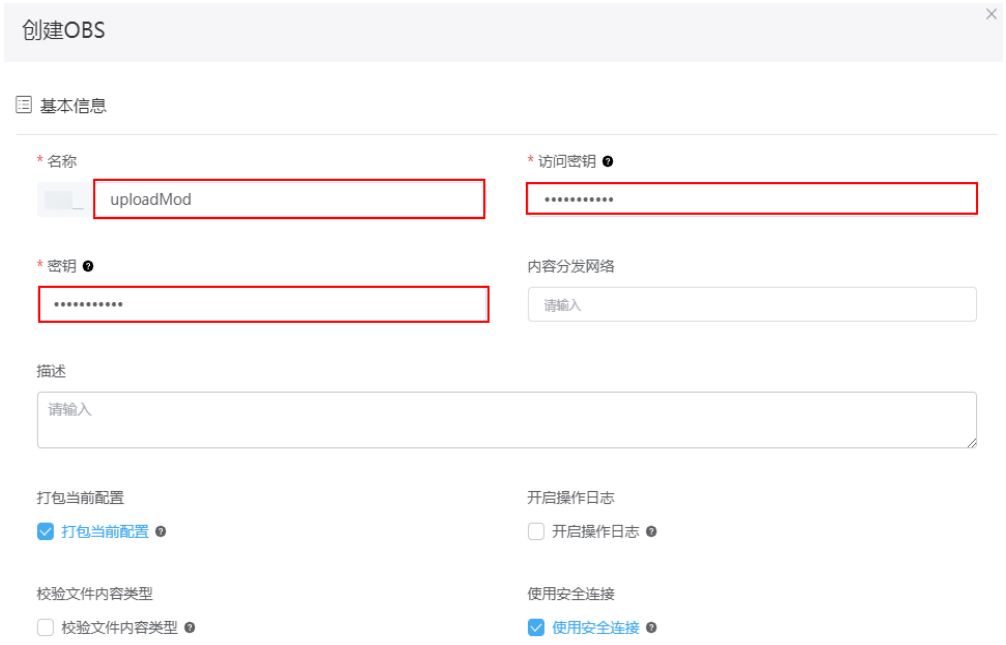


图 9-5 添加桶

表 9-2 新建 OBS 连接器参数说明

参数	说明	示例
名称	新建OBS连接器的名称。命名要求如下： – 长度不能超过64个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	uploadMod
访问密钥	配置用户的访问密钥AK。	**步骤1**中获取的Access Key ID值
密钥	配置与访问密钥AK结合使用的私有访问密钥SK。	**步骤1**中获取的Secret Access Key值
桶	设置存储桶所在的区域和桶名称。	区域选择“华北-北京四”，桶名称为“appcubecn4”，即**步骤1**中提前创建的桶

步骤4 创建对象orcTry，用于存放图片URL等信息。

1. 在应用设计器的左侧导航栏中，选择“数据”，单击对象中的“+”。
2. 设置对象的名称和唯一标识为“orcTry”，单击“确定”。

图 9-6 创建对象 orcTry

表 9-3 新建 orcTry 对象参数说明

参数	说明	示例
对象名称	新建对象的名称，创建后可修改。 取值范围：1~80个字符。	orcTry
唯一标识	新建对象在系统中的标识，创建后不支持修改。命名要求如下： • 长度不能超过63个字符，包括前缀命名空间的长度。 • 必须以英文字母开头，只能由英文字母，数字和下划线组成，且不能以下划线结尾。	orcTry

3. 在已创建的对象中，单击，进入对象详情页面。
4. 在“字段”页签，单击“添加”，为对象添加pictureName字段。

图 9-7 添加 pictureName 字段

表 9-4 添加字段参数说明

参数	说明	示例
显示名称	新建字段的名称，创建后可修改。 取值范围：1~63个字符。	pictureName
唯一标识	新建字段在系统中的标识，创建后不支持修改。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	pictureName
字段类型	单击 ***，在弹出的页面中，根据页面提供的参数解释，选择新建字段所属的类型。	文本

5. 按照上述操作，为对象添加表9-5中字段。

表 9-5 添加其他字段

显示名称	唯一标识	字段类型
pictureId	pictureId	文本

显示名称	唯一标识	字段类型
picUrl	picUrl	文本区
result	result	文本区

步骤5 创建OCR连接器。

1. 在应用设计器中，选择“集成”，单击“连接器”下的“连接器实例”。
2. 在AI中，选择“OCR”，单击“+”，进入创建OCR页面。
3. 输入基本信息，单击“保存”。

图 9-8 设置基本信息

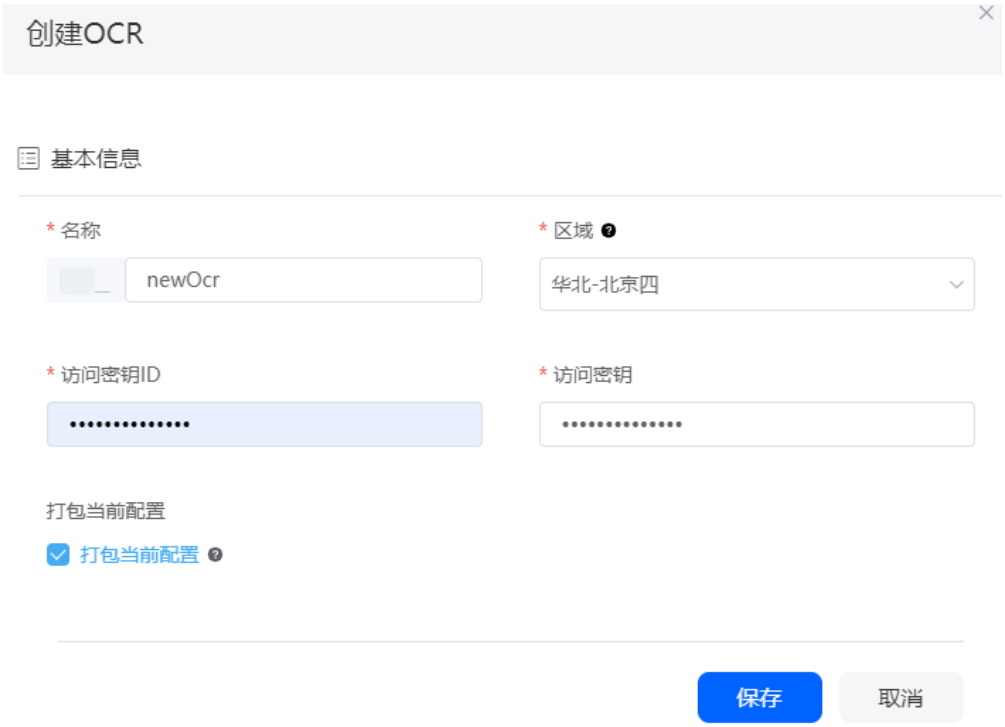


表 9-6 新建 OCR 连接器参数说明

参数	说明	示例
名称	新建OCR连接器的名称。命名要求如下： – 长度不能超过63个字符，包括前缀命名空间的长度。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	newOcr

参数	说明	示例
区域	OCR服务的区域。	华北-北京四
访问密钥ID	访问密钥ID和访问密钥一起使用，对请求进行加密签名。	参考**步骤1**中操作获取，配置为“AK”的值
访问密钥	访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。	参考**步骤1**中操作获取，配置为“SK”的值

步骤6 创建脚本，用于提交图片信息以及OCR接口识别图片内容。

1. 在应用设计器中，选择“逻辑”，单击脚本后的“+”。
2. 新建一个空白脚本，脚本名称为octTryTs。

图 9-9 新建 octTryTs 脚本

3. 在脚本编辑器中，输入如下示例代码。

```
//本脚本用于提交图片信息以及调用ocr接口识别图片内容
import * as db from 'db';//导入处理object相关的标准库
import * as context from 'context';//导入上下文相关的标准库
import * as ocr from 'ocr';

//定义入参结构
@action.object({ type: "param" })
export class ActionInput {
    @action.param({ type: 'String', required: true, label: 'String' })
    prold: string;
    @action.param({ type: 'String', required: true, label: 'String' })
    proName: string;
    @action.param({ type: 'Any', required: true, label: 'Any' })
    url: any;
```

```
}
//定义出参结构，出参包含1个参数，workOrder的记录id
@action.object({ type: "param" })
export class ActionOutput {
    @action.param({ type: 'String' })
    id: string;
    @action.param({ type: 'String', required: true, label: 'String' })
    url: string;
}
//使用数据对象命名空间__orcTry__CST
@useObject(['命名空间__orcTry__CST'])
@action.object({ type: "method" })
export class CreateWorkOrder {    //定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input: 'ActionInput', output: 'ActionOutput' })
    public createWorkOrder(input: ActionInput): ActionOutput {
        let cli = ocr.newClient("命名空间__newOcr"); //创建连接器实例
        let out = new ActionOutput();    //新建出参ActionOutput类型的实例，作为返回值
        let error = new Error();    //新建错误类型的实例，用于在发生错误时保存错误信息
        try {
            let url = "https://appcubecn4.obs.cn-north-4.myhuaweicloud.com/" + input.url[0]
['originalUrl']; // 拼接成完整的图片url信息
            let resp = cli.idCardWithURL(url, "front"); // 调用ocr接口，front表示识别身份证正面
            let productData = new Object();
            productData['命名空间__pictureName__CST'] = input.proName;  //将入参赋值给productData变
量，方便后面使用
            productData['命名空间__pictureId__CST'] = input.proId;
            productData['命名空间__picUrl__CST'] = url;
            productData['命名空间__result__CST'] = JSON.stringify(resp['result']);
            let s = db.object('BJ4__orcTry__CST');    //获取命名空间__orcTry__CST这个Object的操作实例
            let id = s.insert(productData);
            if (id) {
                out.id = id;
                out.url = url;
            } else {
                error.name = "WOERROR";
                error.message = "Unable to create pic!";
                throw error;
            }
        } catch (error) {
            console.error(error.name, error.message);
            context.setError(error.name, error.message);
        }
        return out;
    }
}
```

4. 单击  ，保存脚本，保存成功后单击  ，激活脚本。

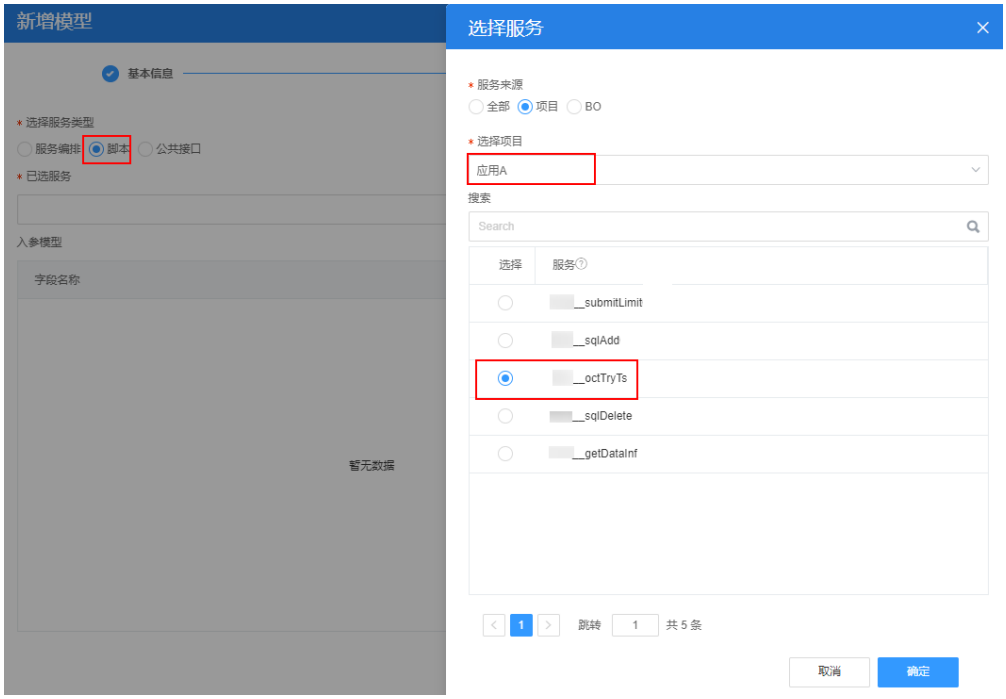
步骤7 新建页面模型。

1. 在应用设计器中，选择“界面”，单击页面后的“+”，新建一个标准页面。
2. 在标准页面底部，单击“模型视图”。
3. 单击“新增模型”，输入模型名称（如orcNew）、“来源”选择“服务”，单击“下一步”。

图 9-10 新建模型

4. 选择步骤6中创建的脚本，单击“确定”后，再单击“下一步”。

图 9-11 选择脚本

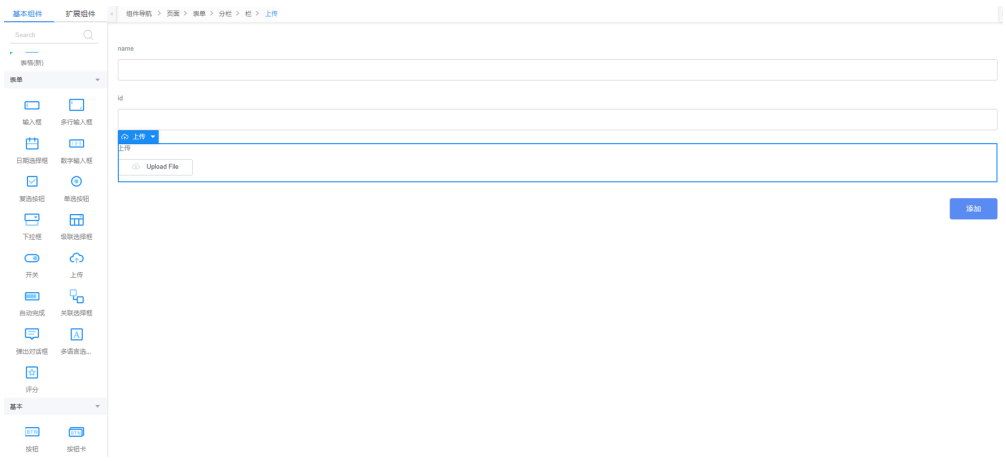


5. 直接单击“确定”，完成模型的创建。

步骤8 创建图片上传表单页面。

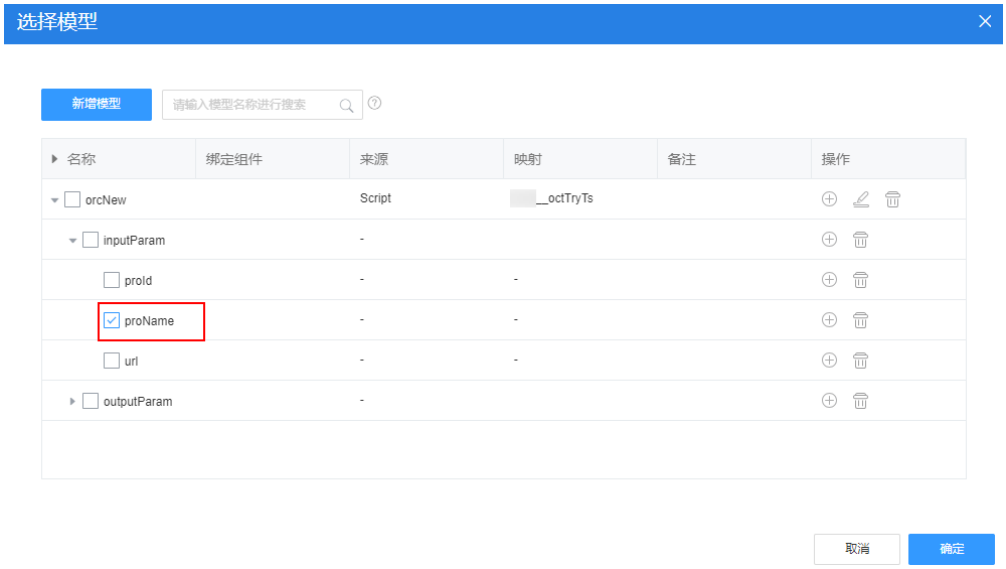
1. 在标准页面中，拖入2个输入框组件、1个按钮和1个上传组件。
2. 将输入框标签分别修改为“name”和“id”，将按钮显示名称设置为“添加”。

图 9-12 表单最终设置效果



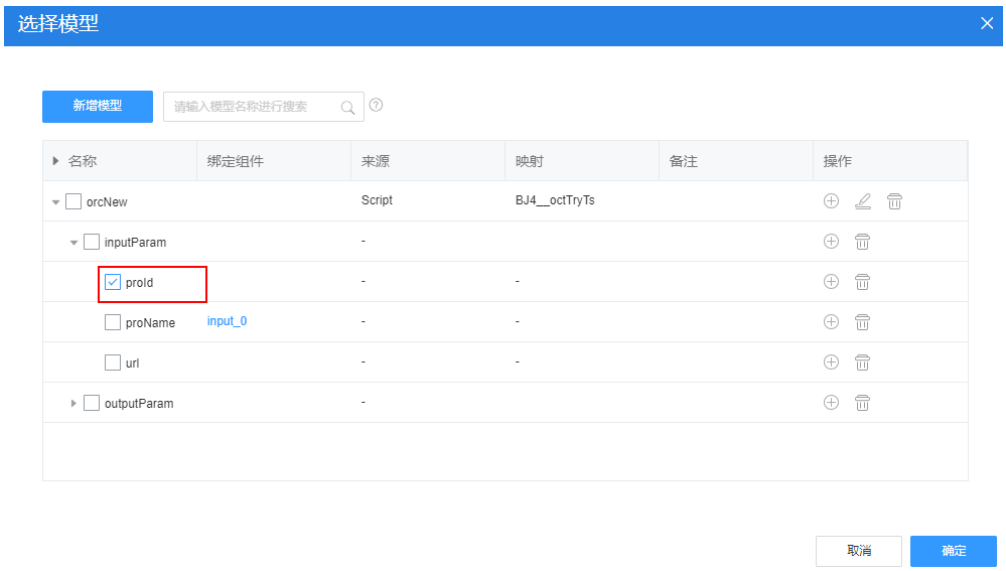
3. 选中name输入框，在“属性 > 数据绑定 > 值绑定”中，单击 。
4. 选择**步骤7**中创建模型中的proName字段，单击“确定”。

图 9-13 选择 proName 字段



5. 选中id输入框，在“属性 > 数据绑定 > 值绑定”中，单击 。
6. 选择**步骤7**中创建模型中的proId字段，单击“确定”。

图 9-14 选择 proId



7. 同样，按照上述操作，为上传组件绑定模型，并设置存储信息。

图 9-15 为上传组件绑定模型

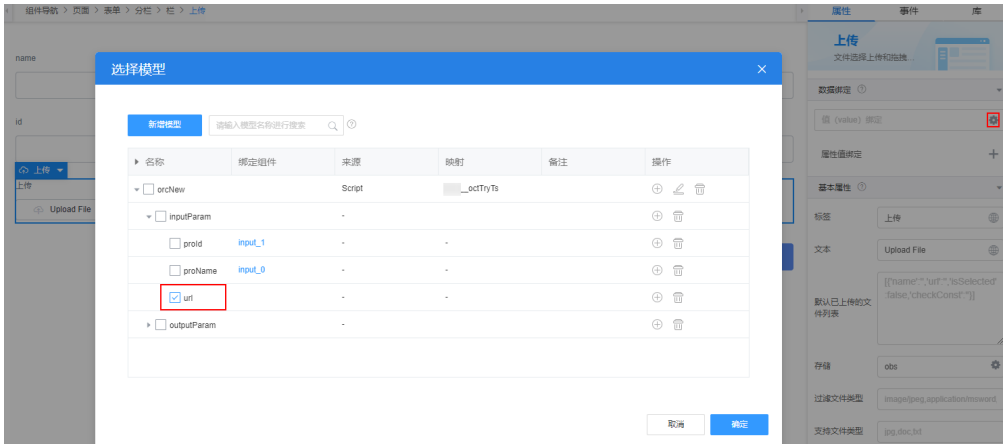
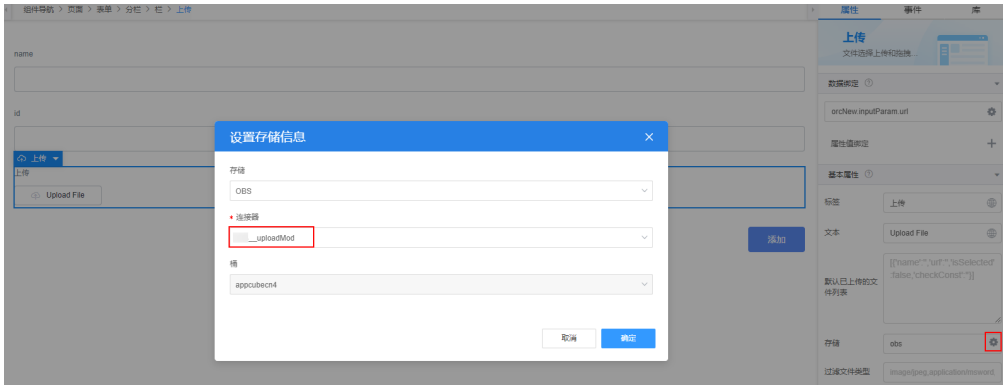


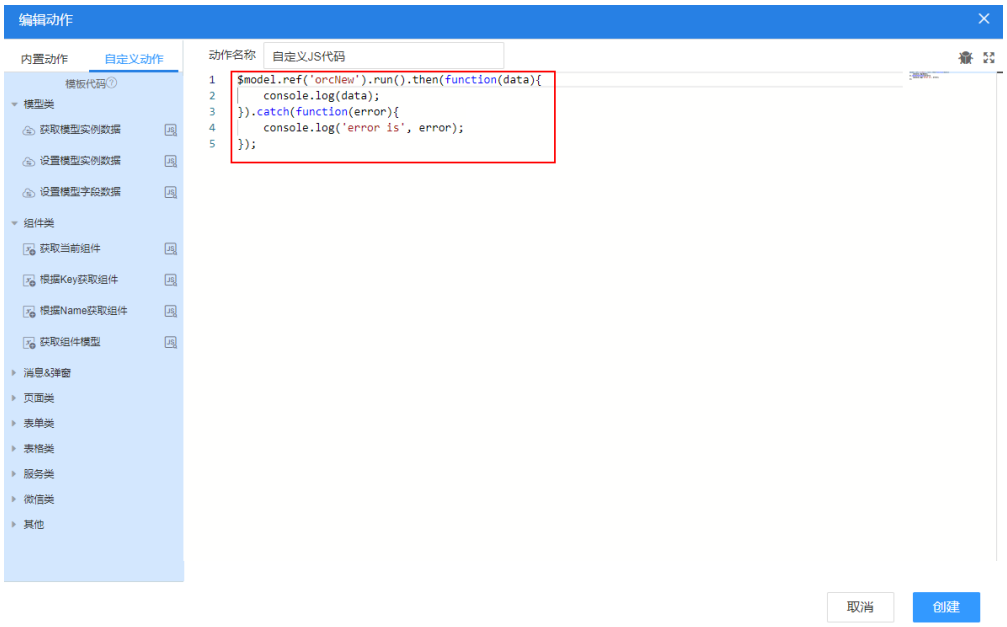
图 9-16 设置存储信息



步骤9 为添加按钮，添加事件。

1. 选中添加按钮组件，选择“事件”页签。
2. 单击“点击”后的“+”，进入添加动作页面。
3. 在自定义动作中，输入自定义代码，单击“保存”。

图 9-17 自定义动作



其中，“orcNew”为步骤7中创建模型的名称。

```
$model.ref('orcNew').run().then(function(data){
    console.log(data);
}).catch(function(error){
    console.log('error is', error);
});
```

步骤10 返回标准页面，单击  ，保存页面，保存成功后单击  ，预览效果。

----结束

9.2 通过 AstroZero 中的连接器实现文件上传功能

期望实现效果

AstroZero封装了不同类型的连接器用于对接其他服务，对接后即可在应用中使用该服务。例如，通过连接器对接OBS后，可将前端页面中上传的文件存储到OBS桶中。

图 9-18 在 OBS 桶的指定路径下可查看到文件

功能实现方法

步骤1 准备工作。

- 需要拥有一个华为账号或一个可用于访问OBS的IAM用户，即先注册华为云并实名认证、创建IAM用户、充值以及购买资源包，具体操作请参见**使用OBS前需要做的准备工作**。
- 获取AK（Access Key ID）、SK（Secret Access Key），即访问密钥对，具体操作请参见**获取AK/SK**。
- 已在华为OBS上，创建存储桶（例如“bing.testonly.1”），用于后续存储对象使用，具体操作请参见**如何创建桶**，请记录创建桶时选择的区域。

步骤2 创建一个低代码应用。

1. 参考**授权用户使用AstroZero并购买实例**中操作，申请AstroZero免费试用或购买商业实例。
2. 实例购买后，在AstroZero服务控制台的“主页”中，单击“进入首页”，进入应用开发页面。
3. 在“应用”中，单击“新建低代码应用”或单击，进入新建低代码应用页面。
4. 在新建低代码应用页面，应用类型选择“标准应用”，单击“确定”。
5. 输入应用的标签和名称，单击“新建”，即可进入应用设计器。

图 9-19 创建一个空白应用

表 9-7 新建空白应用参数说明

参数	说明	示例
标签	新建应用的标签，长度不能超过80个字符。标签是应用在系统中的唯一标识，创建后不支持修改。	我的第一个应用

参数	说明	示例
名称	新建应用的名称，输入标签值后单击该参数的输入框，系统会自动生成应用的名称，同时自动在名称前，添加***命名空间***__。命名要求如下： – 长度不能超过31个字符，包括前缀命名空间的长度。 名称前的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母、数字或单下划线组成，且不允许以下划线结尾。	A

步骤3 创建一个OBS连接器。

1. 在应用设计器中，选择“集成”，单击“连接器”下的“连接器实例”。
2. 在存储中，选择“OBS”，单击“+”，进入创建OBS页面。
3. 输入基本信息并添加桶，单击“保存”。

图 9-20 设置桶基本信息

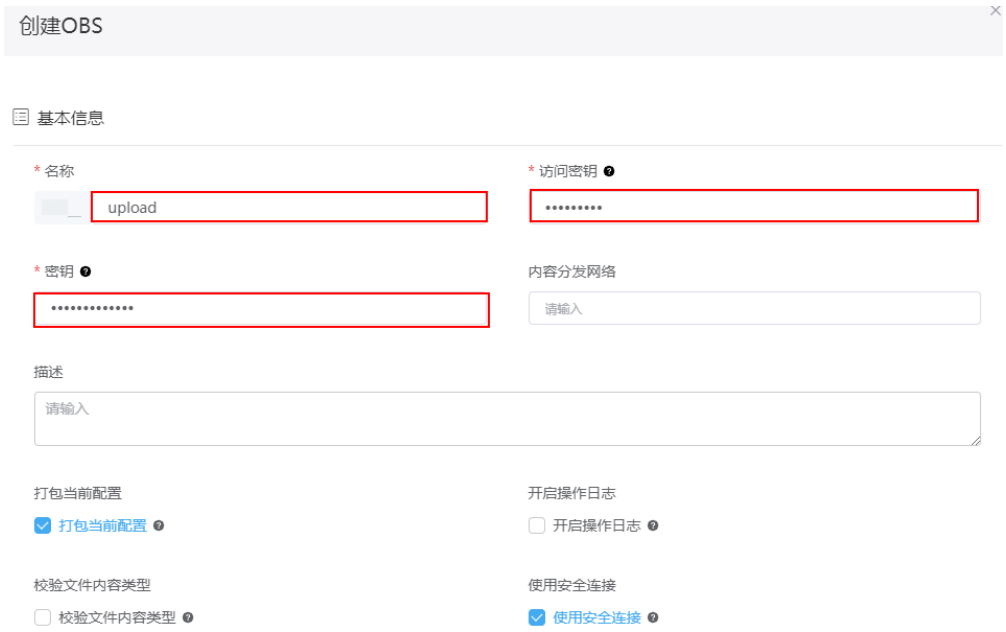


图 9-21 添加桶

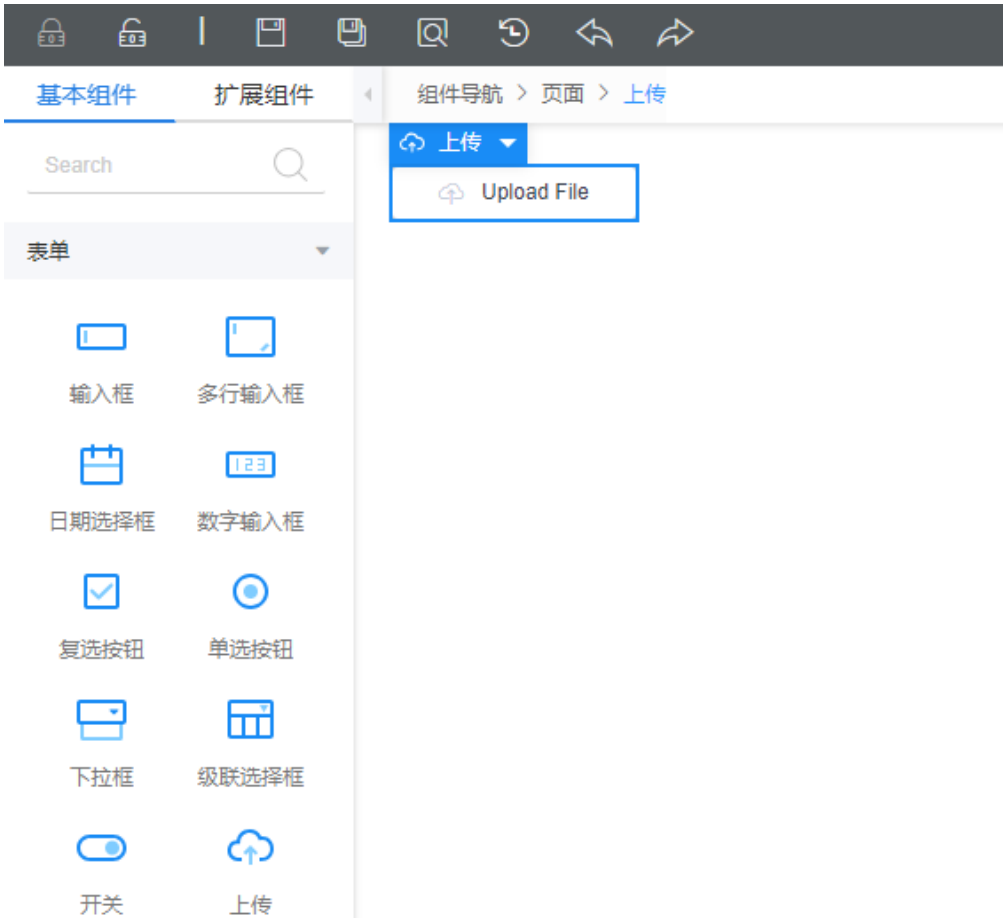
表 9-8 新建 OBS 连接器参数说明

参数	说明	示例
名称	新建OBS连接器的名称。命名要求如下： – 长度不能超过64个字符，包括前缀命名空间的长度。 标识前模糊掉的内容为命名空间，在AstroZero中为了避免不同租户间数据的重名，租户在首次创建应用时需要先定义一个命名空间。一个租户只能创建一个命名空间，创建后不支持修改。 – 必须以英文字母开头，只能由英文字母，数字和单下划线组成，且不能以下划线结尾。	upload
访问密钥	配置用户的访问密钥AK。	**步骤1**中获取的Access Key ID值
密钥	配置与访问密钥AK结合使用的私有访问密钥SK。	**步骤1**中获取的Secret Access Key值
桶	设置存储桶所在的区域和桶名称。	区域选择“华北-北京四”，桶名称为“bing.testonly.1”，即**步骤1**中提前创建的桶

步骤4 创建标准页面，用于上传文件。

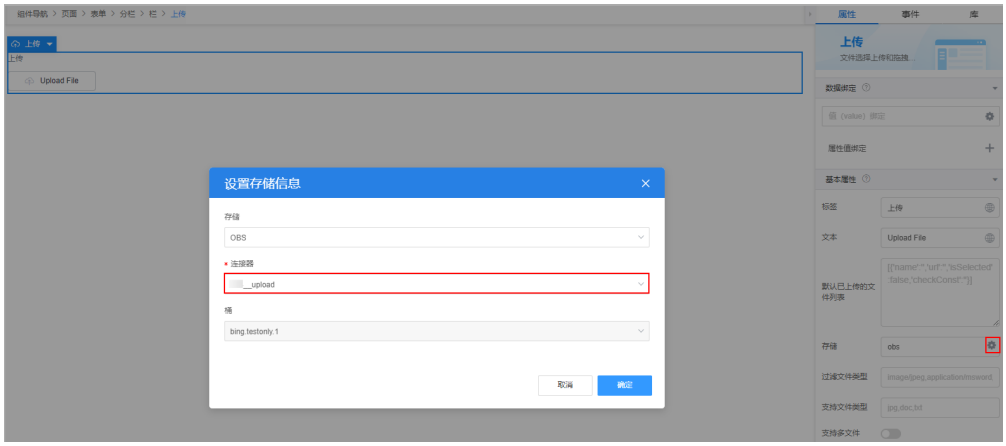
1. 在应用设计器中，选择“界面”，单击页面后的“+”，新建一个标准页面。
2. 在标准页面中，拖入一个“上传”组件。

图 9-22 拖入上传组件



3. 选中上传组件，存储选择“OBS”，并选择**步骤3**中创建好的连接器。

图 9-23 设置存储信息



4. 定义文件的上传路径。

图 9-24 设置上传路径

步骤5 返回标准页面，单击，保存页面，保存成功后单击，预览效果。

----结束

10 移动应用专项

10.1 将 AstroZero 中的应用发布成 WeLink 轻应用

方案概述

在AstroZero中，支持扫码绑定WeLink。绑定WeLink后，可将AstroZero中开发的应用快速发布到WeLink中，实现企业业务的高效率、低成本创新。

发布WeLink轻应用是将已开发好的应用或全新开发的Web页面发布到WeLink，用户在WeLink APP中即可打开应用。发布WeLink轻应用适用于简单的应用场景。下面以创建“调查问卷轻应用”为例，介绍如何把AstroZero上开发的应用发布成WeLink（蓝标）轻应用。

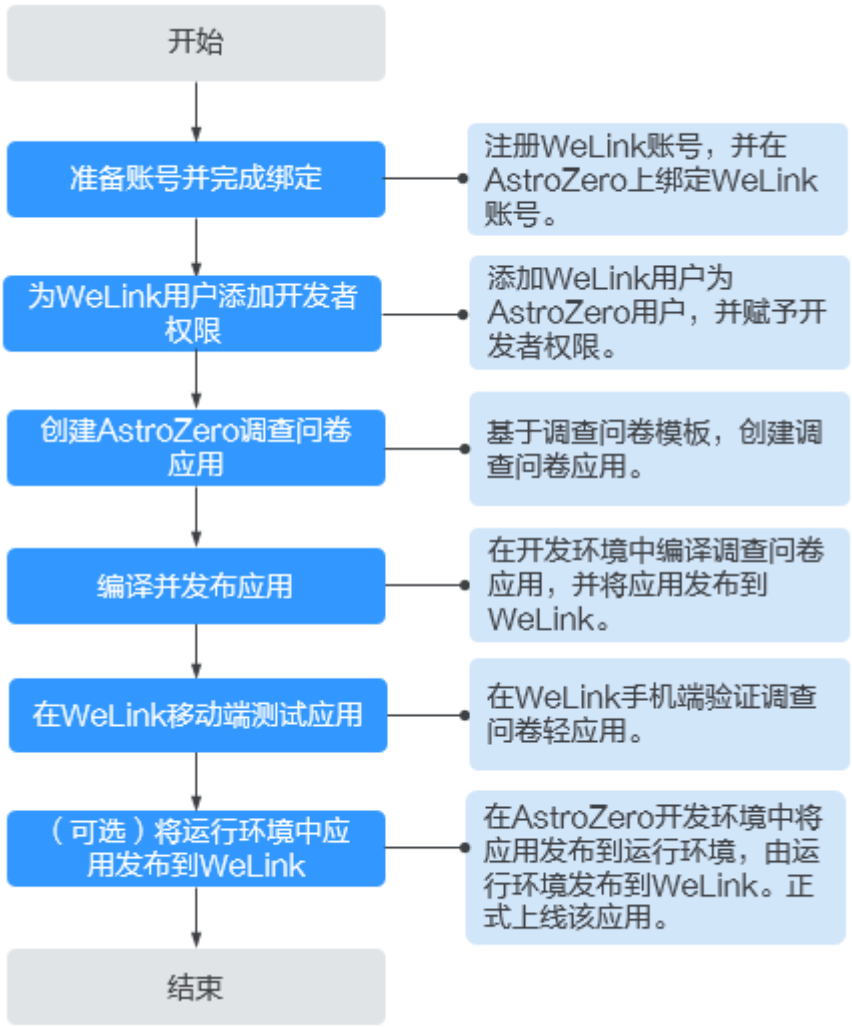
调查问卷模板为企业和个人提供调查问卷应用模板，用于收集资料或管理问题记录。该模板以对AstroZero开发者展开调查问卷为例，可基于该应用模板快速自定义调查问卷内容。

图 10-1 开发者调查问卷

操作流程

将AstroZero中的应用发布成WeLink轻应用的流程，如图10-2所示。

图 10-2 开发 WeLink 轻应用流程



1. **步骤一：准备账号并完成绑定**

   在AstroZero的环境配置中，将AstroZero和WeLink进行绑定。

2. **步骤二：添加WeLink用户为AstroZero开发者**

   如果需要WeLink用户在AstroZero开发环境中具备开发者权限，请添加WeLink用户为AstroZero开发者用户，并赋予开发者权限。

3. **步骤三：创建AstroZero调查问卷应用**

   在AstroZero开发环境中创建“调查问卷应用”，设置应用在移动端显示效果，并为Welink用户设置业务访问权限。

4. **步骤四：编译发布应用**

   将在AstroZero上创建的应用发布到WeLink上。

5. **步骤五：在WeLink移动端测试应用**

   应用发布后，可以在WeLink手机端搜索并验证已发布的轻应用，也可以在企业WeLink管理员界面直接扫码进入轻应用。

6. **步骤6：（可选）将运行环境中应用发布到WeLink**

   如果您购买的是AstroZero商用版实例，还支持将AstroZero开发环境中的应用发布到运行环境，由运行环境发布到WeLink。

步骤一：准备账号并完成绑定

在AstroZero上完成和WeLink账号的绑定操作。

步骤1 注册WeLink账号。

说明

如果已有WeLink账号，请直接执行**步骤2**。

1. 登录**WeLink注册页面**，输入用户名和验证码，单击“下一步”。

图 10-3 WeLink 登录页面

2. 选择待创建WeLink账号的角色，请根据实际需求设置，本示例选择“我是管理员”。

图 10-4 选择待创建 WeLink 账号的角色

说明

如果注册的手机号已加入企业或者组织，则不会显示图10-4页面。

3. 在“创建企业/组织”页面，设置如下信息后，单击“创建”。

图 10-5 创建企业/组织

- 企业/组织名：新建企业或组织名称。
- 姓名：新建企业或组织的联系人。
- 邮箱：新建企业或组织的联系人的邮箱地址。
- 行业：新建企业或组织所属的行业。

WeLink账号注册成功后，可根据实际需求，选择对应的操作，如登录企业管理后台，下载Windows客户端等。

图 10-6 选择对应操作

步骤2 以账号登录AstroZero开发环境，绑定WeLink并同步WeLink信息。

说明

此处的账号是指购买AstroZero服务的华为账号。更多AstroZero中用户的介绍，请参见**AstroZero中的用户**。

1. 以华为账号登录华为云官网。
2. 单击，在查找框中搜索“Astro轻应用”，单击查找的结果，进入AstroZero服务控制台。
3. 在实例页面，单击“进入首页”。
4. 在页面左上角，单击，选择“环境管理 > 环境配置”，进入AstroZero环境配置页面。
5. 在左侧导航栏中，选择“集成连接 > 统一身份认证设置”，单击“WeLink”。

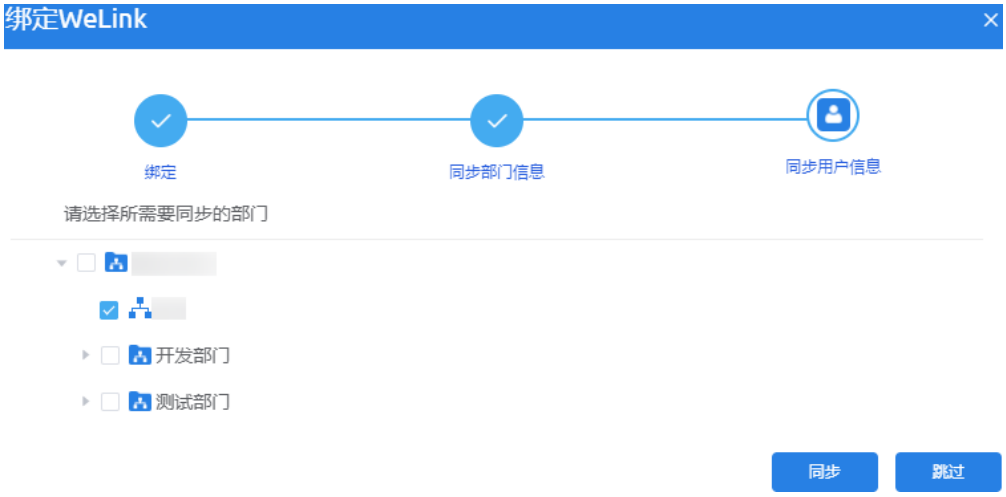
图 10-7 进入绑定 WeLink 页面

6. 在弹出页面，单击“绑定WeLink”。
7. 在弹出的“扫码认证”框中，打开移动端WeLink，在右上角单击“+”，扫描**图10-8**中所示二维码。

图 10-8 扫描二维码

8. 勾选待同步用户所在的WeLink部门，单击“同步”。

图 10-9 同步用户信息



说明

- 绑定WeLink会同步WeLink用户为AstroZero业务用户，用户数不计入License用户数限额。只有业务用户登录当前AstroZero环境时，才会计入License用户数限额。
- AstroZero绑定WeLink时，扫码绑定的WeLink用户会自动添加为AstroZero开发者用户和业务用户。
- 同步WeLink部门时，会自动同步WeLink部门主管为AstroZero业务用户。
- 同步的WeLink用户包括子部门的用户。

勾选要同步用户所在的WeLink部门，系统还会**同步所选部门的WeLink用户**，将WeLink用户同步到AstroZero当前环境的**业务用户**列表中，并赋予“Portal User Profile”权限。在AstroZero环境配置的“维护 > 全局元素 > 业务用户”中，可查看同步来的WeLink用户信息。

思考：如果在图10-9中未同步用户信息，即单击“跳过”，完成WeLink账号绑定后，后续该如何同步用户信息呢?

在AstroZero环境配置的“集成连接 > 统一身份认证设置”，单击“WeLink”，在“同步用户”区域下单击“立即同步”，即可完成用户信息的同步。

----结束

步骤二：添加 WeLink 用户为 AstroZero 开发者

添加WeLink用户为AstroZero开发者用户，并赋予开发者权限。

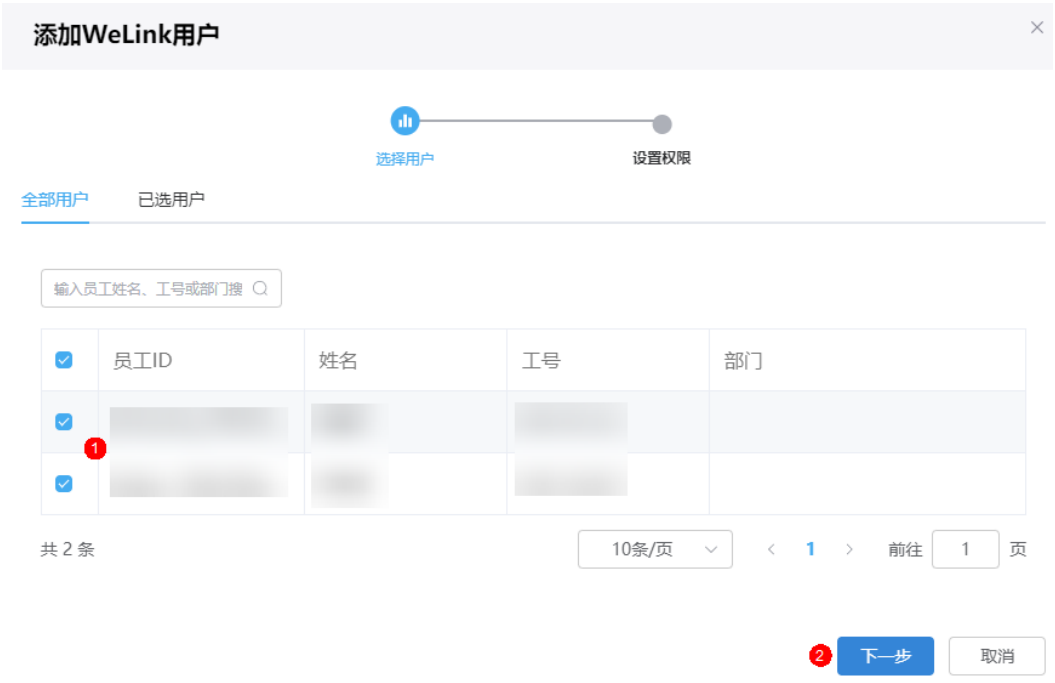
步骤1 以华为账号登录AstroZero环境配置。

步骤2 在左侧导航栏中，选择“用户安全 > 用户”，单击“添加WeLink用户”。

图 10-10 添加 WeLink 用户

步骤3 勾选需要添加的WeLink用户，单击“下一步”。

图 10-11 勾选 WeLink 用户



步骤4 权限配置为“Developer Profile”，单击“保存”。

添加完成后，在用户列表中可查看到已添加的WeLink用户。

WeLink用户添加了开发者权限后，即可使用WeLink扫码登录AstroZero开发环境，具体操作如下：

1. 登录**WeLink管理后台**。

图 10-12 登录页面

2. 打开移动端WeLink，在右上角单击“+”，扫描二维码进行登录，也可单击**图10-12**上的，使用WeLink用户的账号密码登录。

----结束

步骤三：创建 AstroZero 调查问卷应用

在AstroZero开发环境中创建“调查问卷应用”，设置应用在移动端显示效果，并为Welink用户设置业务访问权限。

步骤1 基于调查问卷模板，创建调查问卷应用。

1. 使用具备开发者权限的AstroZero账号，登录华为云。
2. 单击，在查找框中搜索“Astro轻应用”，单击查找的结果，进入AstroZero服务控制台。
3. 在实例页面，单击“进入首页”，进入AstroZero应用开发页面。

说明

具备开发者权限的AstroZero账号：AstroZero租户及具备开发者权限的WeLink用户，都具有进入开发环境开发应用的权限。

4. 在应用开发页面，选择“模板中心”。
5. 在低代码专区，找到“调查问卷”模板，并单击该模板。

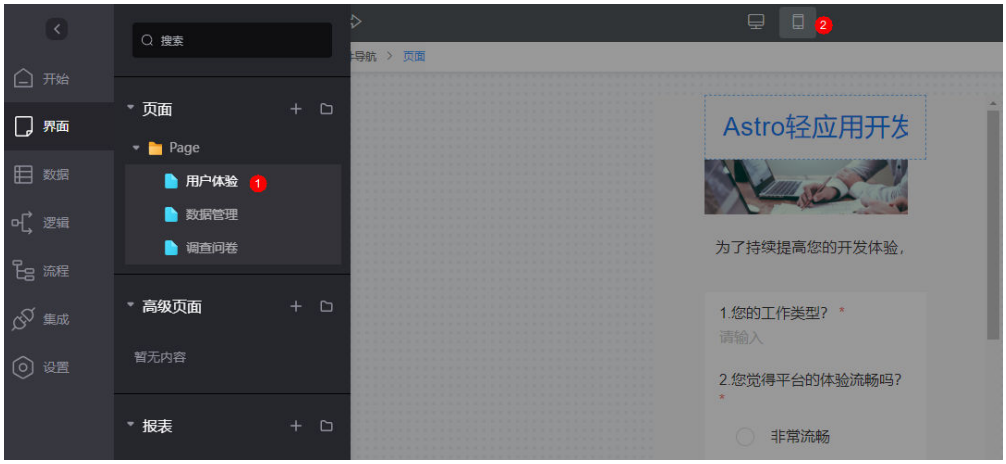
图 10-13 使用应用模板创建应用

6. 在模板详情页面，单击“安装模板”。
   安装完成后，自动进入调查问卷应用设计器。

图 10-14 调查问卷应用

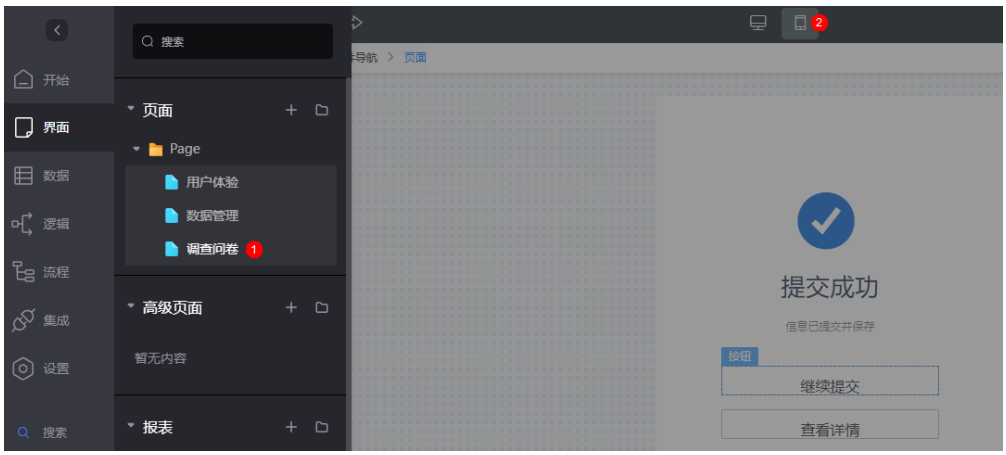
7. 单击调查问卷后的  ，选择“修改应用标签和描述”，将应用标签设置为“开发者调查问卷轻应用”，单击“保存”。
8. 修改要发布的页面模式，单击“Page”目录下的“用户体验”页面，单击页面  ，切换页面布局模式为手机模式，单击  。

图 10-15 修改用户体验布局为手机模式



9. 按照同样的方式，修改页面“调查问卷”的布局模式为手机模式。

图 10-16 修改调查问卷布局为手机模式



10. 在主菜单中，单击“运行 > 立即运行”，可预览调查问卷应用。

步骤2 设置调查问卷应用在移动端显示效果。

1. 在左侧导航栏中，单击“设置”。
2. 在“移动应用设置”中，开启小程序模式。

图 10-17 移动应用设置

步骤3 配置允许WeLink用户使用业务应用权限。

1. 进入AstroZero环境配置，在左侧导航栏中，选择“用户安全 > 权限”。
2. 在“权限配置列表”中，单击“Portal User Profile”。
3. 在权限配置详情中，单击“应用程序设置”页签，单击，选中该应用的“可见性”、“默认”，单击，配置应用可见性。

图 10-18 配置应用可见性

4. 在权限配置详情中，单击“自定义对象权限”页签，进入编辑页面。搜索“Questionnaires_qR__CST”对象，在搜索结果中找到“*命名空间*__Questionnaires_qR__CST”，单击，先勾选“修改全部”，再勾选“创建”权限，单击保存配置。
   执行此操作的目的是，允许WeLink用户操作当前应用中的自定义对象。

图 10-19 配置自定义对象的操作权限

----结束

步骤四：编译发布应用

本节介绍如何将在AstroZero上创建的应用发布到WeLink上。

步骤1 在AstroZero上编译发布应用，将创建的应用编译发布到WeLink上。

1. 进入Astro轻应用开发者调查问卷设计器。
2. 在主菜单中，选择“发布 > 应用打包 > 生成移动应用”。
3. 在发布应用中，单击“WeLink-轻应用”。

图 10-20 WeLink-轻应用

4. 打开WeLink，扫描登录WeLink开放平台。

图 10-21 扫描登录 WeLink 开放平台

步骤2 在WeLink上提交发布应用申请。

1. 在“应用开发 > 首页 > 轻应用 > 版本管理”中，单击“发布版本”。

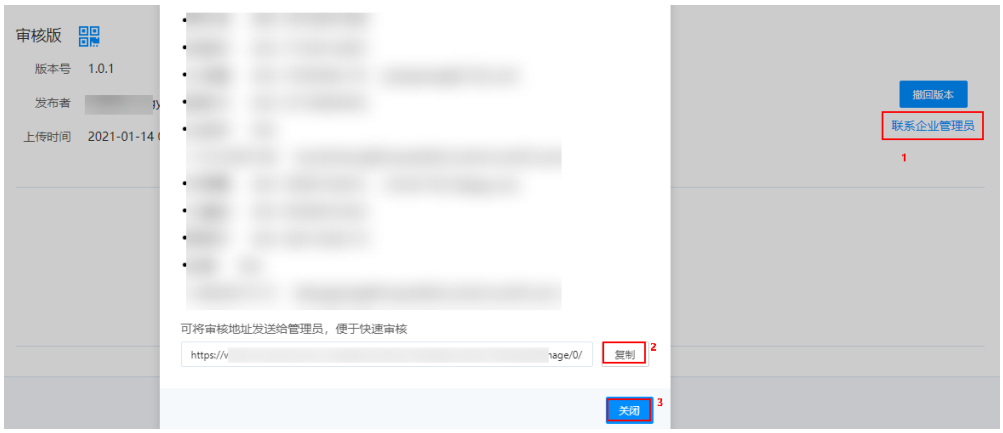
图 10-22 在 WeLink 上发布版本

2. 选择审核员，在“版本说明”中输入说明信息，如“第一次测试发布”，单击“确定”，提交发布应用申请给审核员审核。

图 10-23 输入版本说明

3. 提交审核后，页面将显示“审核版”，单击“联系企业管理员”，在弹窗中单击“复制”，可将应用发布审核链接发送给审核员审核发布版本，单击“关闭”。

图 10-24 联系管理员审核版本



步骤3 联系审核员，审核发布应用版本。

1. 审核员打开应用发布审核链接，登录后，单击“审核新版本”。

   如果是应用测试阶段，可设置部分测试人员可见，再单击“同意”。如果需要直接使用开发环境发布到WeLink中的轻应用作为业务应用正式上线，可设置全部人员可见。

图 10-25 同意发布

2. 在“同意新版本”中，输入审核意见，单击“确定”，完成审核。
   审核完成后，应用会发布到企业WeLink中。

----结束

步骤五：在 WeLink 移动端测试应用

应用发布后，可以在WeLink手机端搜索并验证已发布的轻应用，也可以在企业WeLink管理员界面直接扫码进入轻应用。

步骤1 在WeLink手机端，单击“业务”，搜索应用“开发者调查问卷轻应用”。

步骤2 搜索到后，单击应用名，进入调查问卷页面。

图 10-26 搜索轻应用“开发者调查问卷轻应用”

步骤3 输入问卷信息，单击“提交”，成功后跳转到“提交成功”页面。

图 10-27 进入轻应用页面

图 10-28 提交成功

问卷调查完成后，切回到“开发者调查问卷”的应用开发页面，单击顶部菜单栏中的“运行 >立即运行”，预览应用。在“调查问卷管理”页签中，可查看或删除相关数据。

图 10-29 管理调查问卷记录

AstroZero提供了很多模板应用供您直接使用，方便您快速将应用发布到WeLink中，省去开发环节。WeLink用户可直接在WeLink移动端使用该应用。如果租户购买的AstroZero规格为专业版或专享版，拥有运行环境，您可继续执行**步骤6：（可选）将运行环境中应用发布到WeLink**，使用运行环境将应用发布到WeLink，WeLink用户在WeLink移动端使用该应用时，调用的后台接口服务都承载在运行环境中。

----结束

步骤 6：（可选）将运行环境中应用发布到 WeLink

在AstroZero开发环境中将应用发布到运行环境，由运行环境发布到WeLink。

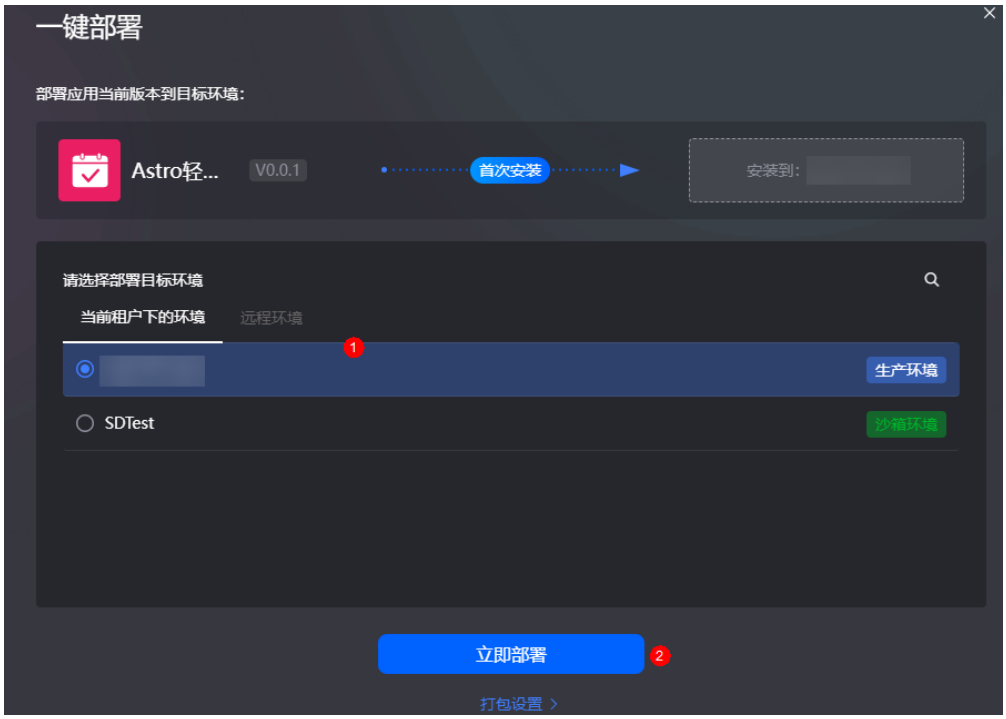
说明

免费版不提供运行环境，请购买专业版或专享版实例后，再进行打包发布操作，并在运行环境安装应用。

步骤1 将应用发布到运行环境。

1. 进入AstroZero应用开发页面，在“主页 > 全部应用”中，单击“Astro轻应用开发者调查问卷”后的“编辑”，进入应用设计器。
2. 在应用设计器的顶部菜单栏中，单击“发布 > 一键部署”。
3. 选择目标运行环境，单击“立即部署”。

图 10-30 一键部署



4. 在弹出的确认对话框中，单击“立即部署”。

   在应用设计器的右下角，可查看应用的部署进度。应用部署完成后，单击图10-32中的“立即访问”，即可访问该应用。

图 10-31 查看部署进度

图 10-32 访问应用

步骤2 在AstroZero运行环境部署应用。

1. 在AstroZero已购买的实例中，单击“管理运行环境”，进入运行环境。
2. 在运行环境的右上方，单击“体验新版”，进入新版运行环境。
3. 在左侧导航栏中，选择“应用管理 > 应用列表”，在应用列表中可查看到已安装的调查问卷应用。

图 10-33 查看已安装的应用

步骤3 在WeLink上创建正式的轻应用，并参考**步骤2**，在AstroZero运行环境的“集成管理 > 统一身份认证设置”中，单击“WeLink”，再单击“绑定WeLink”。

步骤4 配置同步到AstroZero的WeLink用户允许使用该业务应用权限。

1. 在AstroZero运行环境的左侧导航栏中，选择“用户安全 > 权限”。
2. 在“权限配置列表”中，单击“Portal User Profile”。
3. 在权限配置详情中，单击“应用程序设置”页签，单击，选中该应用的“可见性”复选框，单击。配置应用可见性。

图 10-34 配置应用可见性

4. 在权限配置详情中，单击“自定义对象权限”页签，进入编辑页面。搜索“Questionnaires__qR__CST”对象，在搜索结果中找到“*命名空间*__Questionnaires_qR__CST”，单击，先勾选“修改全部”，再勾选“创建”

权限，单击，保存配置。执行此操作的目录是，允许WeLink用户操作当前应用中的自定义对象。

图 10-35 配置自定义对象的操作权限

说明

实际的对象名，可在“Astro轻应用开发者调查问卷”应用开发页面的“Model”目录下查看。

步骤5 在运行环境将应用发布到WeLink。

1. 在运行环境中，选择“应用管理 > 应用列表”，在应用程序列表中单击待发布应用“*命名空间*__Questionnaires”后的，选择“WeLink-轻应用”，发布应用到WeLink上创建的正式应用中。

图 10-36 选择 WeLink-轻应用

2. 参考**步骤2~步骤3**，在WeLink上发布版本并审核，审核时设置使用该应用的所有WeLink用户可见。

   应用发布后，使用该应用的所有WeLink用户可以在WeLink手机端搜索并使用已发布的轻应用。第三方网站也可通过WeLink移动端扫码登录应用，详情请参见**扫码登录第三方网站集成指导**。

----结束

常见问题

- **问题1**

  问题描述:

  将AstroZero和WeLink账号A绑定后，如果需要绑定另外的WeLink账号B，怎么处理?

  处理方法:

  需要先将AstroZero和WeLink账号A解绑，再绑定其他WeLink账号。

a. 在AstroZero当前环境中，选择“集成连接 > 统一身份认证 ”，单击“WeLink”，再单击“解绑”。

图 10-37 解绑 WeLink

b. 在弹出的提示框，根据情况勾选“删除部门和用户信息”，表示是否删除已同步的部门和用户信息，单击“确定”。

c. 选择“集成连接 > 统一身份认证”，单击“WeLink”，再单击“绑定WeLink”，绑定其他WeLink账号。

- **问题2**

  问题描述：

  WeLink导入的业务用户，在PC端无法正常登录已发布的应用。

  可能原因：

  在AstroZero中导入业务用户时，未同步WeLink中的业务用户密码。

  处理方法：

  请参考**扫码登录第三方网站集成指导**中操作处理。

10.2 将 AstroZero 中的应用发布成 WeLink We 码应用

方案概述

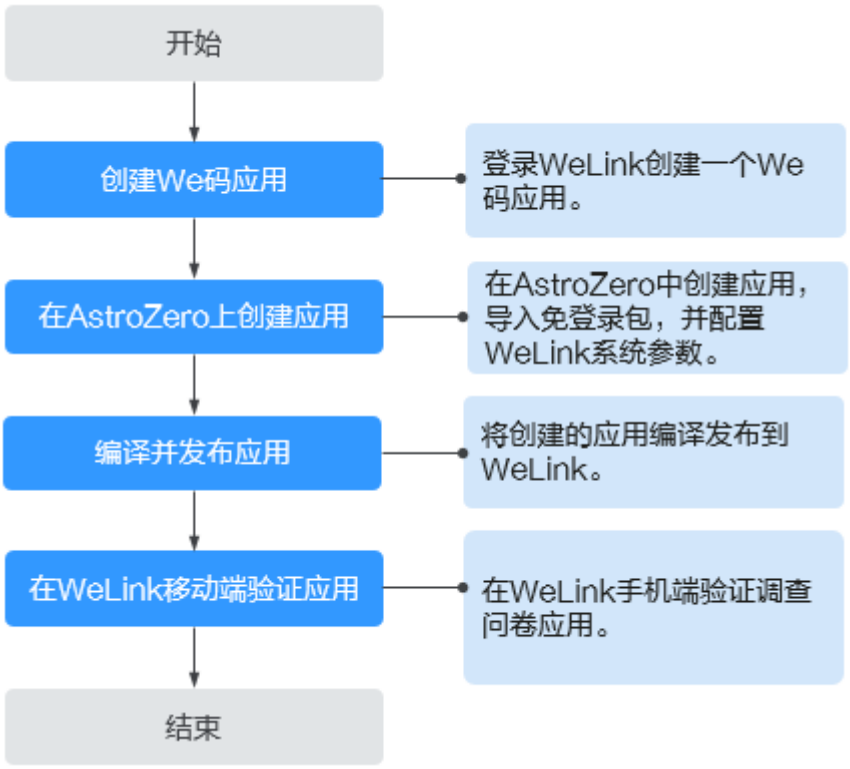
AstroZero上开发的应用可以发布成WeLink（蓝标）We码应用。We码是将前端的静态资源放入WeLink，都是本地访问，适用于对页面加载速度要求比较高的场景。关于WeLink We码应用的详细介绍，可参见**WeLink开发之旅**。

以创建“调查问卷应用”为例，向您介绍如何将AstroZero中的应用发布成We码应用。

操作流程

将AstroZero中的应用发布成WeLink We码应用的流程，如**图10-38**所示。

图 10-38 We 码应用发布流程



1. **步骤一：创建We码应用**

   在WeLink端发布小程序之前，需要以WeLink管理员账号登录WeLink开发平台，创建一个We码应用。

2. **步骤二：在AstroZero上创建应用**

   We码应用创建完成后，需要在AstroZero租户账号中，创建一个应用，导入一个免登录App包，再配置企业WeLink相关系统参数。

3. **步骤三：编译并发布应用**

   将已创建的应用编译发布到WeLink上，并设置AstroZero端及WeLink端的应用发布参数。

4. **步骤四：在WeLink移动端测试应用**

   应用发布后，在WeLink移动端搜索并验证已发布的小程序应用。

步骤一：创建 We 码应用

在WeLink端发布小程序之前，需要以WeLink管理员账号登录WeLink开发平台，创建一个We码应用。

步骤1 打开浏览器，访问**WeLink开放平台**。

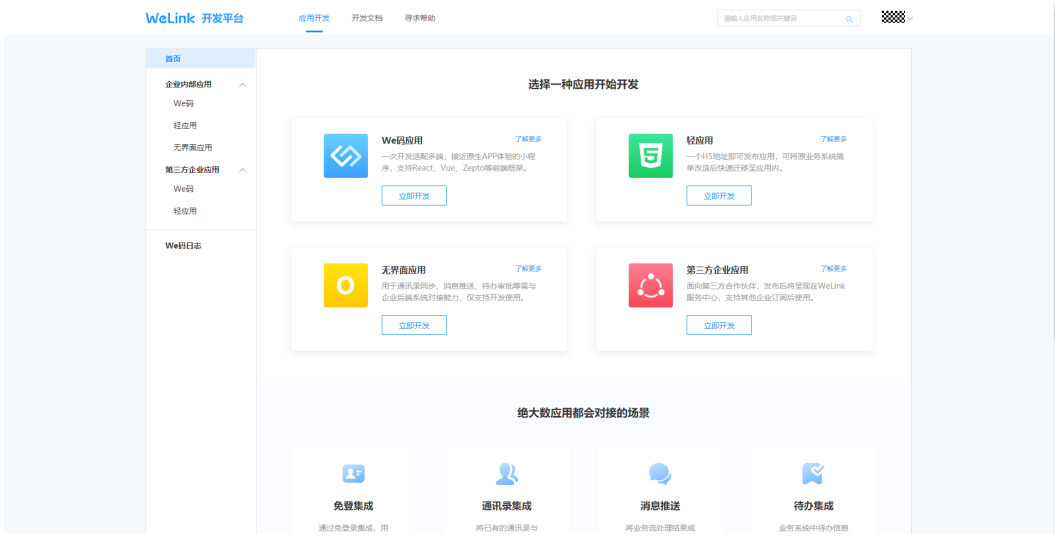
图 10-39 访问 WeLink 开放平台

步骤2 在图10-39中单击右上角用户名后的 ，选择“进入开发者后台”。

说明

如果使用已有账号直接登录WeLink开放平台，此处请选择“登录开发者后台”，并通过WeLink移动端扫码或输入WeLink开发者账号密码登录。

图 10-40 登录开发者后台



步骤3 在“We码应用”下，单击“立即开发”，进入创建We码应用页面。

图 10-41 We 码管理

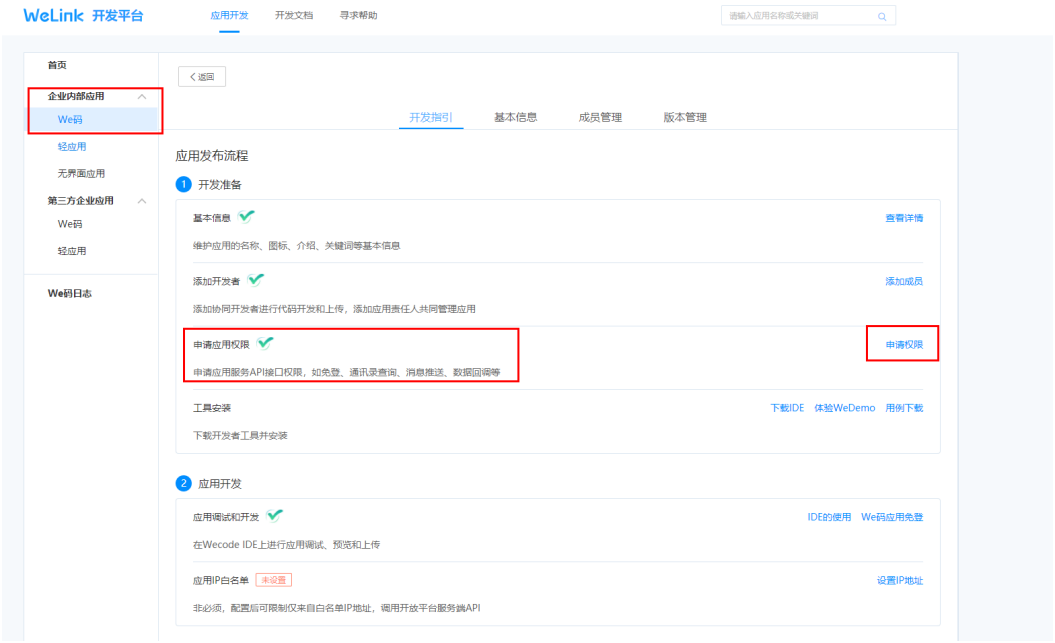
步骤4 在创建We码应用页面，设置应用相关信息，单击“提交”。

图 10-42 创建 We 码

- 图标：上传新建We码应用的图标，也可直接使用推荐的图标。
- 小程序中文名称：本示例配置为“开发者调查问卷”。
- 小程序英文名称：本示例配置为“MyWelinkApp”。

步骤5 选择“企业内部应用 > We码 > 开发指引”，在“开发准备”中单击“申请应用权限”右侧的“申请权限”。

图 10-43 登录 WeLink 开发平台



步骤6 在申请权限页面，分别单击“用户登录信息”、“轻应用鉴权”、“租户详细信息”、“组织架构信息”和“用户详细信息”后的“申请权限”，申请相关权限。

图 10-44 申请相关权限

身份与鉴权

权限名称	权限描述	状态	操作
用户登录信息	获取用户的基本信息，用于登录系统和应用	未开通	申请授权
轻应用鉴权	轻应用鉴权，用于调用JS-API	未开通	申请授权

通讯录

权限名称	权限描述	状态	操作
租户详细信息	获取租户详细信息，如租户名称、许可、规模等	未开通	申请授权
组织架构信息	按userid或部门id，以部门编码方式获取组织架构	未开通	申请授权
用户简单信息	按部门id获取用户简单信息，如userid、工号、中文名、英文名	未开通	申请授权
用户详细信息	按userid或部门id获取用户更多详细信息，如工号、手机号、邮箱等详细信息	未开通	申请授权
通讯录同步	企业向WeLink同步通讯录信息	未开通	申请授权
用户邮箱信息	获取用户信息，包括邮箱等敏感信息	未开通	申请授权
部门人员信息	根据部门编码查询部门人员信息	未开通	申请授权
用户手机信息	获取用户信息，包含手机号、邮箱等敏感信息	未开通	申请授权
用户权限信息	获取用户权限信息，包括分域管理员所管理的部门范围等	未开通	申请授权

说明

如果要发布的应用调用了WeLink的其他接口，请添加其他相关权限。示例中的应用只开启上述权限。

步骤7 在“基本信息”页签，记录应用的唯一标识“client_id”和应用密钥“client_secret”，用于接口鉴权。

图 10-45 查看应用的唯一标识和密钥

----结束

步骤二：在 AstroZero 上创建应用

We码应用创建完成后，需要在AstroZero中创建一个应用，导入一个免登录App包，再配置企业WeLink相关系统参数。

步骤1 基于调查问卷模板，创建调查问卷应用。

1. 使用具备开发者权限的AstroZero账号，登录华为云官网。
2. 单击☰，在查找框中搜索“Astro轻应用”，单击查找的结果，进入AstroZero服务控制台。
3. 在实例页面，单击“进入首页”，进入AstroZero应用开发页面。

   说明

   - 具备开发者权限的AstroZero账号：AstroZero租户及具备开发者权限的WeLink用户，都具有进入开发环境开发应用的权限。
   - 租户请使用华为账号登录AstroZero，具备开发者权限的WeLink用户则需要访问**AstroZero产品页**扫码登录。
4. 在应用开发页面，选择“模板中心”。
5. 在低代码专区，找到“调查问卷”模板，并单击该模板。

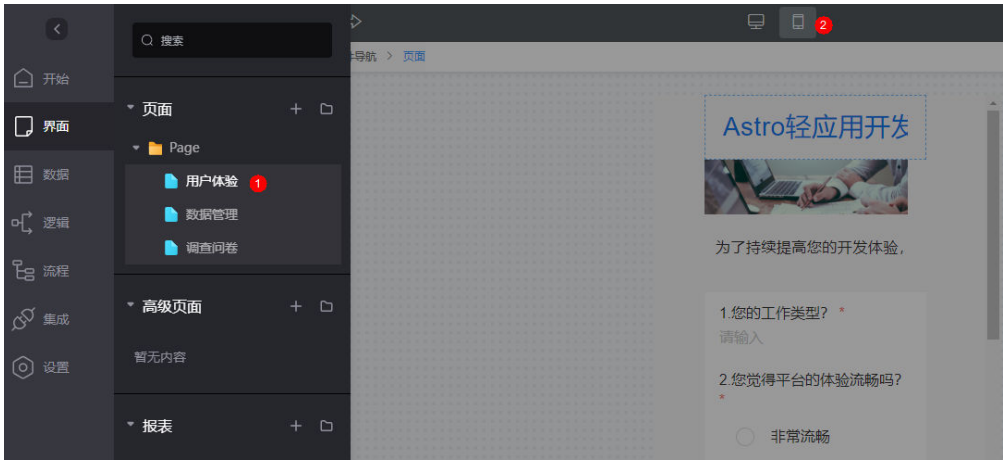
图 10-46 使用应用模板创建应用

6. 在模板详情页面，单击“安装模板”。
   安装完成后，自动进入调查问卷应用设计器。

图 10-47 调查问卷应用

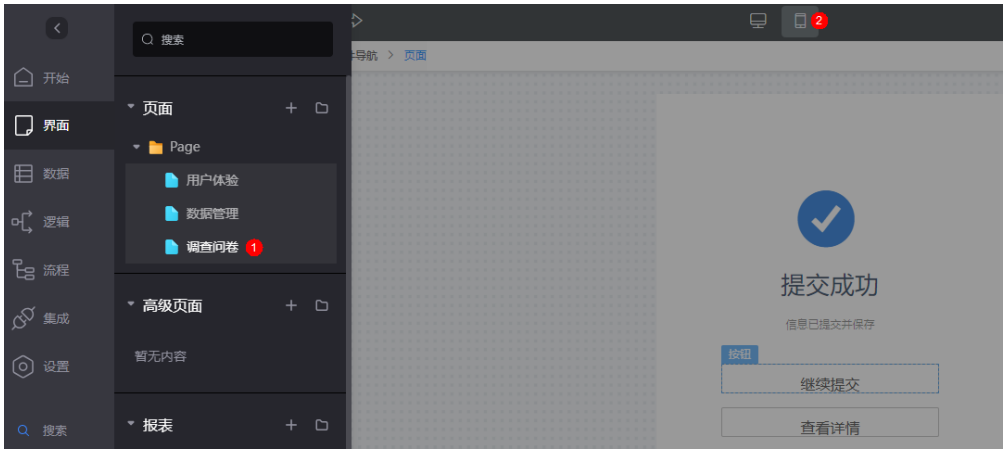
7. 单击调查问卷后的  ，选择“修改应用标签和描述”，将应用标签设置为“Astro轻应用开发者调查问卷”，单击“保存”。
8. 修改要发布的页面模式，单击“Page”目录下的“用户体验”页面，单击页面  ，切换页面布局模式为手机模式，单击  。

图 10-48 修改用户体验布局为手机模式



9. 按照同样的方式，修改页面“调查问卷”的布局模式为手机模式。

图 10-49 修改调查问卷布局为手机模式



10. 在主菜单中，单击“运行 > 立即运行”，可预览调查问卷应用。

步骤2 导入免登录应用包。

说明

免登录应用包仅为样例包，用于调查问卷应用发布到WeLink We码应用场景，实现WeLink用户在使用We码应用时，可通过免登逻辑获取到AstroZero的认证信息。开发者可根据实际业务需求自定义开发免登逻辑，入参出参需要和样例保持一致，确保免登逻辑正常。

1. 单击**下载链接**，下载免登录应用包。
2. 在AstroZero应用开发页面，单击，选择“环境管理 > 环境配置”，进入环境配置页面。
3. 在左侧导航栏中，选择“应用管理 > 安装管理 > 包安装”。
4. 单击“新建”，将免登录包拖拽到上传文件处，单击“安装”。

图 10-50 导入软件包

5. 返回AstroZero应用开发页面，在“应用 > 全部”中，可查看到“WeLink”应用。

图 10-51 查看已安装应用

步骤3 配置WeLink相关系统参数及匿名用户权限。

1. 在AstroZero环境配置页面的左侧导航栏中，选择“系统设置 > 系统参数”。
2. 在系统参数列表中，查找参数“welinkClientId”，单击查找到的“welinkClientId”，进入参数详情页面。
3. 单击“值”后的，将“值”修改为**步骤7**中获取的应用唯一标识“client_id”取值，单击“保存”。

图 10-52 修改“值”为应用标识

4. 按照上述方法，将参数“welinkClientSecret”的“值”修改为**步骤7**中，获取的应用密钥“client_secret”取值。

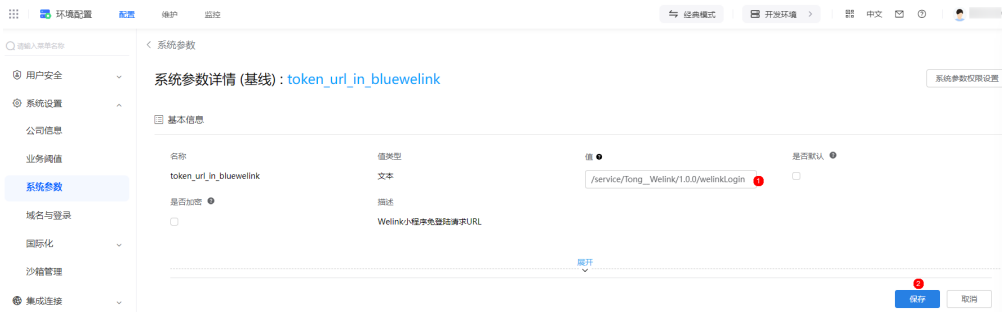
说明

“welinkClientId”和“welinkClientSecret”参数只需要在开发第一个We码应用时进行配置，后续开发其他We码应用不需要再次配置。

5. 修改WeLink小程序回调参数。

在“内置系统参数”页签，搜索“token_url_in_bluewelink”，修改“值”为“/service/Tong__Welink/1.0.0/welinkLogin”。

图 10-53 修改内置系统参数值用于小程序回调



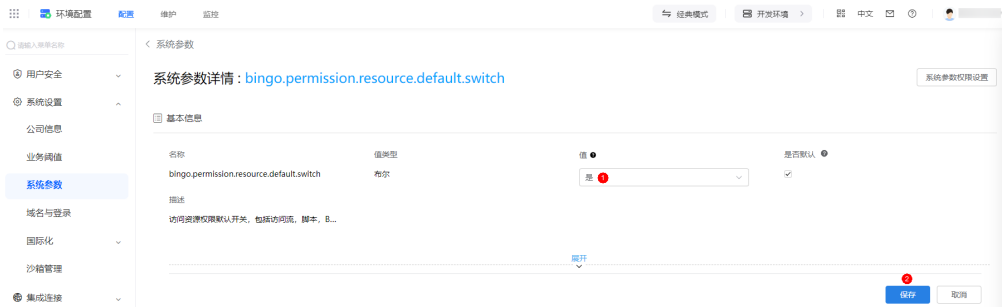
说明

“/service/Tong__Welink/1.0.0/welinkLogin”是免登录应用WeLink中“welinkLogin”接口的URL。设置该“值”，后续将应用发布到Welink后，AstroZero自动将WeLink用户添加到AstroZero业务用户列表中，并归属到Portal User Profile权限集。

6. 开启接口配置权限。

   在“内置系统参数”页签，搜索“bingo.permission.resource.default.switch”，将参数的值修改为“是”。

图 10-54 开启接口权限



7. 配置允许WeLink用户访问小程序权限。

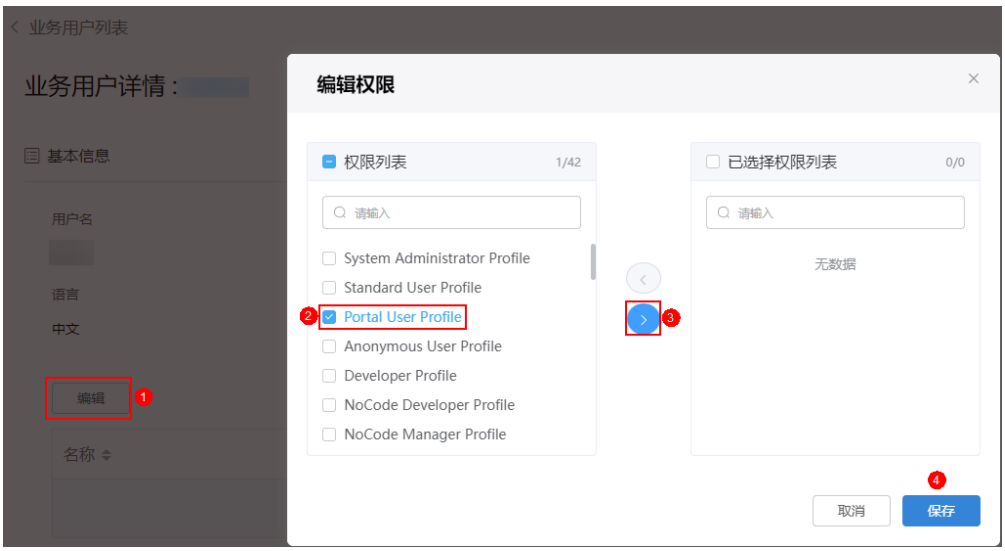
   选择“用户安全 > 权限”，在“权限配置列表”中，单击“Anonymous User Profile”，在“业务权限凭证”页签，单击  ，勾选“welinkAccess”后的“可接入”。

图 10-55 匿名用户权限配置

8. 给WeLink用户配置权限。

   选择“维护 > 全局元素 > 业务用户”，在业务用户列表中，单击WeLink用户名，参考图10-56在业务用户详情中，单击“编辑”，赋予WeLink用户“Portal User Profile”权限。

图 10-56 配置权限



9. 配置应用可见性。

   选择“用户安全 > 权限”，在“权限配置列表”中，单击“Portal User Profile”。在权限配置详情中，单击“应用程序设置”页签。单击，选中应用的“可见性”复选框，单击，配置应用可见性。

图 10-57 配置应用可见性

10. 配置允许WeLink用户操作当前应用中的自定义对象权限。

    选择“用户安全 > 权限”，在“权限配置列表”中，单击“Portal User Profile”。在“自定义对象权限”页签，搜索“Questionnaires_qR__CST”对象，在搜索结果中找到“*命名空间*__Questionnaires_qR__CST”，单击，先勾选“编辑”，保存后再勾选其他全部权限。

图 10-58 配置自定义对象的操作权限（编辑权限）

图 10-59 配置自定义对象的操作权限（其他权限）

说明

这里以“调查问卷”模板应用为例配置“Portal User Profile”权限，WeLink用户使用该应用只要配置相关对象权限即可。使用其他应用，可能还需要配置接口、脚本等相关权限，请根据实际情况配置权限。

----结束

步骤三：编译并发布应用

将已创建的应用编译发布到WeLink上，并设置AstroZero端及WeLink端的应用发布参数。

步骤1 在AstroZero上编译发布应用。

1. 进入Astro轻应用开发者调查问卷设计器。
2. 在主菜单中，选择“发布 > 应用打包 > 生成移动应用”。
3. 在发布应用中，单击“WeLink-We码”。

图 10-60 WeLink-We 码

4. 打开WeLink，扫描登录WeLink开放平台。

图 10-61 扫描登录 WeLink 开放平台

说明

如果当前未登录WeLink，需要先扫码或输入账号、密码登录。登录后，直接跳转到应用创建设置页。

步骤2 设置WeLink应用信息。

1. 在“发布至WeLink”页面，设置待发布应用的以下信息。

- 小程序名称：从下拉框选择要发布到WeLink上的应用名。本示例选择“开发者调查问卷”。
- 版本号：要发布的版本号。注意是数字递增的，新发布的版本号应高于之前的版本。本示例设置为“1.0.0”。
- 主页：选择一个发布的页面作为首页，即显示的默认页面。本示例设置为“*命名空间*_Questionnaires_questionnairePage”。
- 环境类型：承载应用程序服务的AstroZero环境类型。本示例选择“开发环境”。
- 开启调试：开启后，在手机端测试小程序时会展示VConsole，用于调试。建议在测试阶段开启，正式使用时关闭。

2. 设置完成后，单击“发布”。
3. 参数确认无误后，单击“确定”。

图 10-62 确定发布后跳转至 WeLink 应用管理页

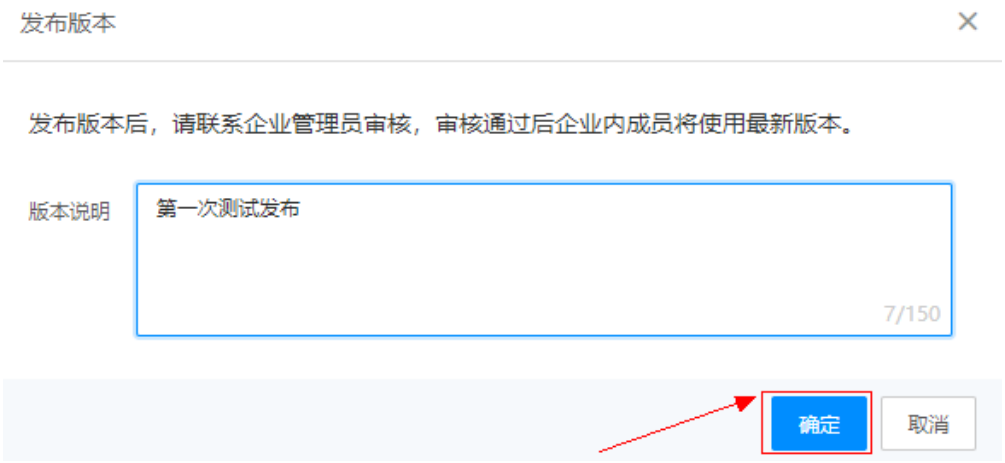
步骤3 在WeLink上提交发布应用申请。

1. 在“应用开发 > 首页 > We码 > 版本管理”中，单击“发布版本”。

图 10-63 在 WeLink 上发布版本

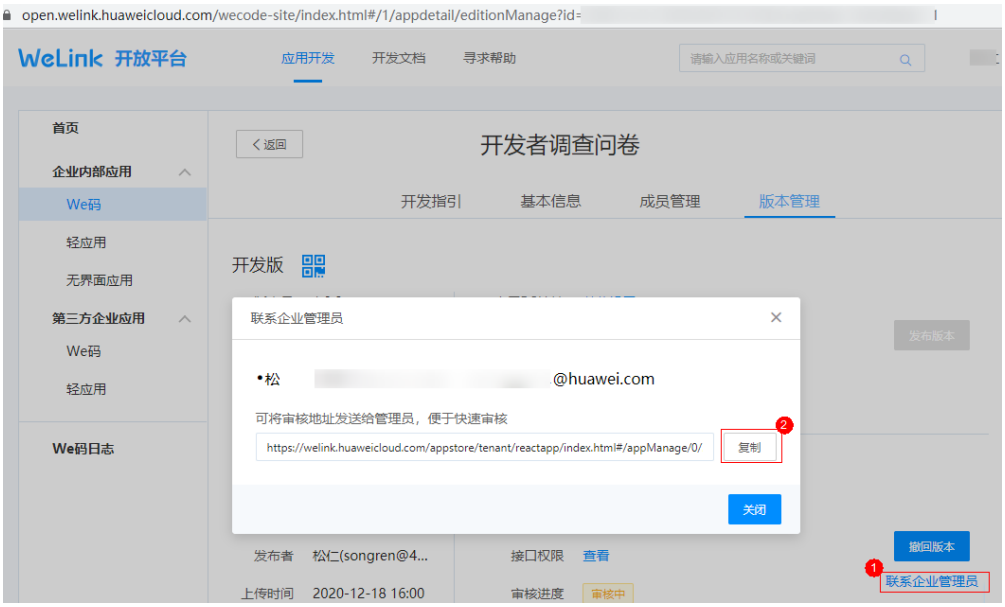
2. 在“版本说明”中输入相关说明信息，如“第一次测试版本”，单击“确定”，提交发布应用申请给企业管理员审核。

图 10-64 输入版本说明



3. 提交审核后，页面将显示“审核版”，单击“联系企业管理员”，在弹窗中单击“复制”，将应用发布审核链接发送给管理员审核发布版本。

图 10-65 联系管理员审核版本



步骤4 联系管理员，审核发布应用版本。

1. 管理员打开应用发布审核链接，登录后，在“审核新版本”中单击“同意”。

   说明

   如果没有应用发布审核链接，请访问**WeLink管理后台**，选择“应用 > 应用管理 > 全部应用”，搜索到小程序，单击应用名即可进入应用审核页面。

2. 在“同意新版本”中，输入审核意见，单击“确定”，完成审核。

   审核完成后，应用会发布到企业WeLink的开发环境中。

----结束

步骤四：在 WeLink 移动端测试应用

应用发布后，可以在WeLink移动端搜索并验证已发布的小程序应用。

步骤1 在WeLink手机端，单击“业务”，搜索小程序“开发者”。

步骤2 单击搜索结果列表中的应用名，进入调查问卷页面。

如果无法正常登录应用，且提示用户名已注册，请按照**常见问题**中操作处理。

图 10-66 搜索小程序“开发者调查问卷”

步骤3 输入问卷信息，单击“提交”，成功后跳转到“提交成功”页面。

图 10-67 进入小程序页面

步骤4 单击“查看详情”或“继续提交”，测试验证小程序的功能。

图 10-68 提交成功

问卷调查完成后，切回到“开发者调查问卷”的应用开发页面，单击 预览应用。在“调查问卷管理”页签中，可查看或删除相关数据。

图 10-69 管理调查问卷记录

----结束

相关主题

如果对导入的免登录应用有兴趣，可以查看以下内容，更深入的了解免登录应用。

- 免登录应用所在位置

  将免登录应用包（**下载链接**）导入到AstroZero后，在应用开发页面“应用 > 全部应用”下会显示导入的应用包。

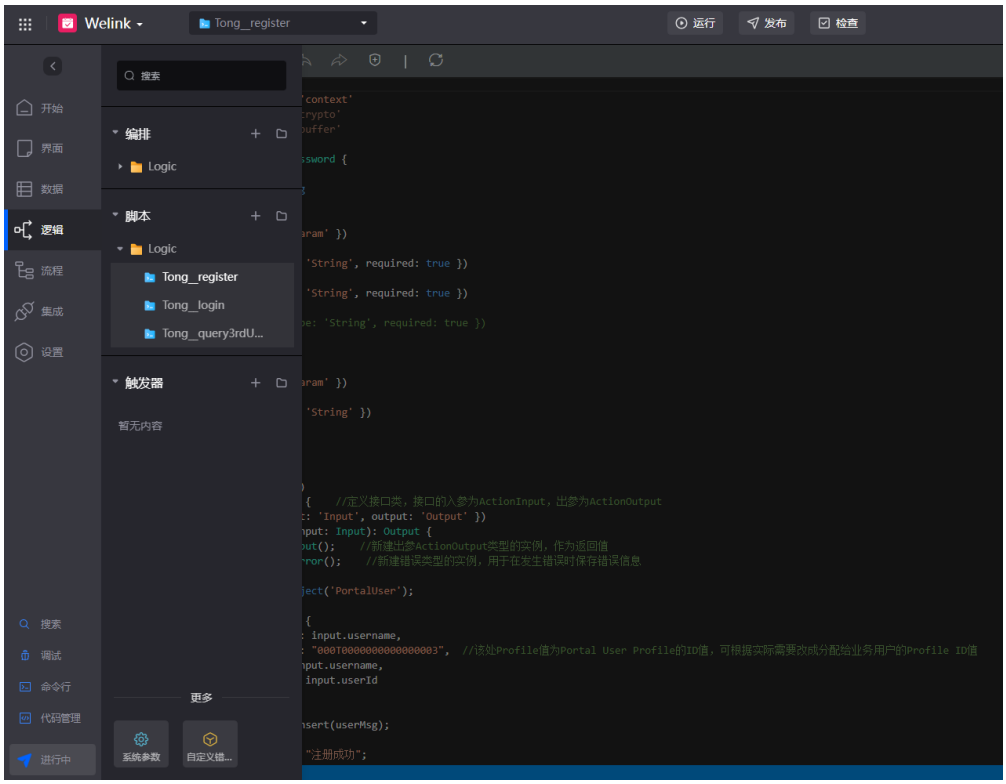
  说明

  本示例中提供的免登录包仅供学习使用，如果需要商用，建议参考样例包进行优化。

图 10-70 免登录应用位置

- 免登录应用开发工作台介绍

图 10-71 应用目录



- 脚本、服务编排和公共接口等应用资源介绍

表 10-1 脚本及接口等应用资源用途

资源名称	所在位置	作用
Tong__register	应用目录Logic	业务用户注册脚本，用于WeLink新用户接入。 **说明** “Tong__”为免登录应用包开发时，开发者使用的命名空间。此命名空间不能修改。
Tong__registerquery3rdUser	应用目录Logic	查询WeLink用户信息脚本，用于WeLink用户接入验证。

资源名称	所在位置	作用
Tong__welinkLogin	应用目录Logic	登录服务编排，用于WeLink用户登录业务应用。
welinkAccess	设置 > 权限设置	业务权限凭证，用于管理外部调用免登录接口的权限。
welinkLogin	集成 > 开放接口	开放接口，用于对外暴露服务，并编辑业务权限凭证，归入welinkAccess。
token_url_in_bluewelink	环境配置 > 系统设置 > 系统参数 > 内置系统参数	AstroZero内置系统参数，用于小程序回调。

- AstroZero应用与企业WeLink互通涉及的接口

  应用发布到WeLink后，WeLink企业中的用户在We码中使用该应用时，AstroZero需要实现WeLink企业用户到应用业务用户的转换，获取用户身份相关接口说明如表10-2所示。

  表 10-2 获取用户身份相关接口

序号	描述	接口	接口类型
1	We码小程序获取免登授权码	HWH5.getAuthCode	前端JS API
2	服务端获取access_token	/auth/v2/tickets	服务端API
3	服务端根据免登授权码获取userId	/auth/v2/userid	服务端API
4	服务端获取用户详细信息	/contact/v1/users	服务端API

常见问题

- **问题1**

  问题描述：在WeLink移动端无法登录应用，且提示“用户名已注册”。

图 10-72 用户名已注册

可能原因：

可能是因为您有多个Welink账号导致的。

处理方法：

a. 在AstroZero应用开发页面，单击，选择“环境管理 > 环境配置”，进入环境配置页面。

b. 在环境配置页面的顶部菜单栏中，选择“维护”。

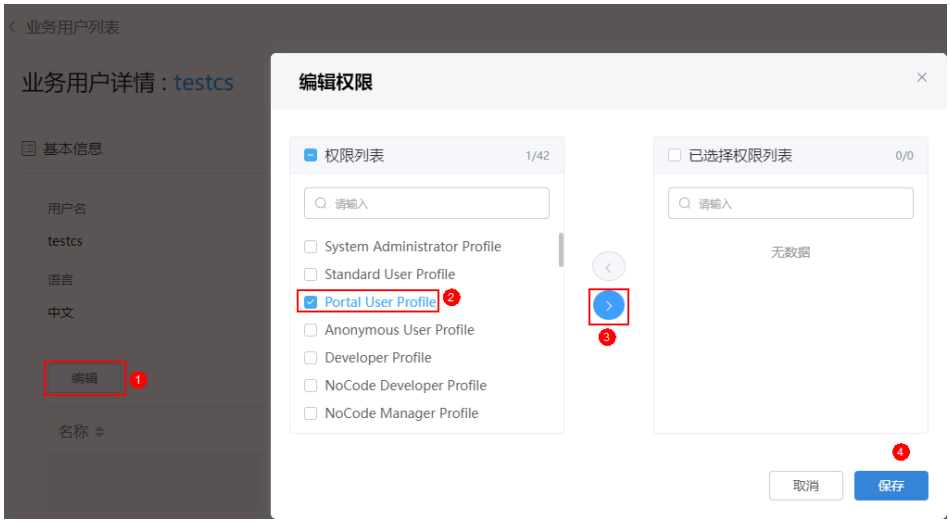
c. 在左侧导航中，选择“全局元素 > 业务用户”。

d. 在业务用户列表中，检查同步过来的Welink账号是否均已自动分配了Portal User Profile权限集。

e. 如果权限集为空，请单击用户名，为此用户添加Portal User Profile权限集，添加完成后，重新打包发布。

图 10-73 多个 welink 账号手动添加 Portal User Profile 权限集

图 10-74 为用户添加 Portal User Profile 权限



- **问题2**

  问题描述：

  WeLink导入的业务用户，在PC端无法正常登录已发布的应用。

  可能原因：

  在AstroZero中导入业务用户时，未同步WeLink中的业务用户密码。

  处理方法：

  请参照**扫码登录第三方网站集成指导**中操作处理。

10.3 将 AstroZero 中的应用发布到微信小程序

方案概述

AstroZero允许将标准页面或高级页面发布到微信小程序。本节将介绍在AstroZero上发布应用到微信小程序的基本操作，包括微信公众平台小程序注册、小程序登录鉴权、发布配置、并验证小程序是否发布成功。

实施步骤

步骤1 准备微信公众平台账号，即在微信公众平台注册小程序。

1. 访问**微信公众平台**，单击右上角“立即注册”。

图 10-75 立即注册

2. 选择注册的账号类型为“小程序”。
3. 请填写邮箱（该邮箱未被微信公众平台注册，未被微信开放平台注册，未被个人微信号绑定），设置登录密码，填写验证码，并同意相关协议，单击“注册”。

   填写的邮箱后续会作为管理员账号进行使用。
4. 登录邮箱，查收激活邮件，单击激活链接。
5. 单击激活链接后，按照界面提示继续下一步的注册流程。

   请选择主体类型，由于该发布功能面向的微信公众账号主体类型只能属于企业、政府、媒体或者其他组织，“主体类型”要选择非个人类型，并完善主体信息和管理员信息。
   - 企业类型账号可选择两种主体验证方式：
     - 方式一：需要用公司的对公账户向腾讯公司打款来验证主体身份。打款信息在提交主体信息后可以查看到。
     - 方式二：通过微信认证验证主体身份，需支付300元认证费。认证通过前，小程序部分功能暂无法使用。
   - 政府、媒体、其他组织类型账号，必须通过微信认证验证主体身份。认证通过前，小程序部分功能暂无法使用。
6. 单击“确定”，完成注册流程。

步骤2 使用管理员账号（即注册过程中填写的邮箱）登录**微信公众平台**，选择“管理 > 成员管理”，在“项目成员”和“体验成员”区域可新增开发者和体验者。

步骤3 使用管理员账号登录**微信公众平台**，获取如下关键信息并配置业务域名和request合法域名。

图 10-76 获取关键信息

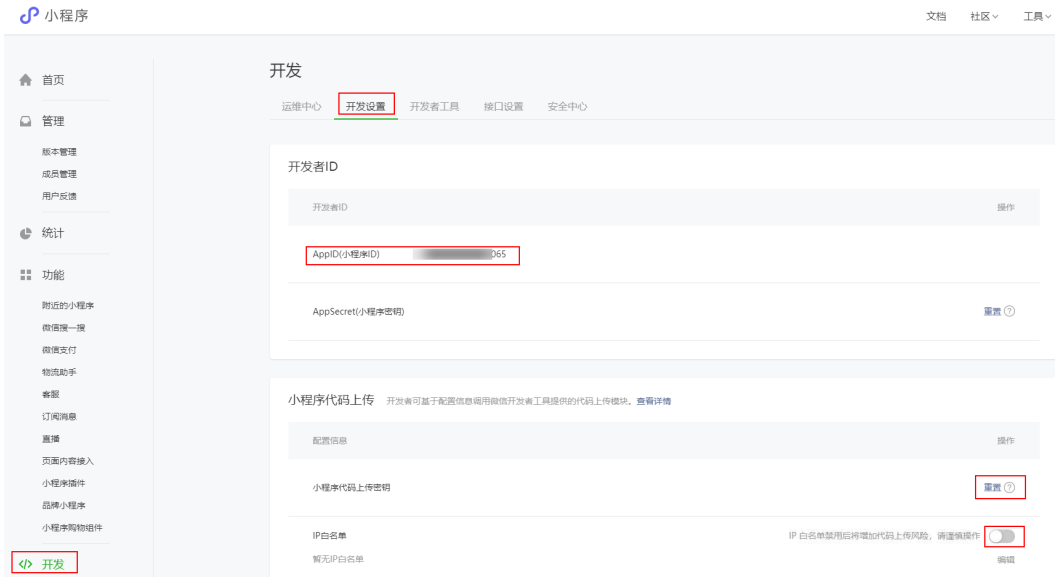
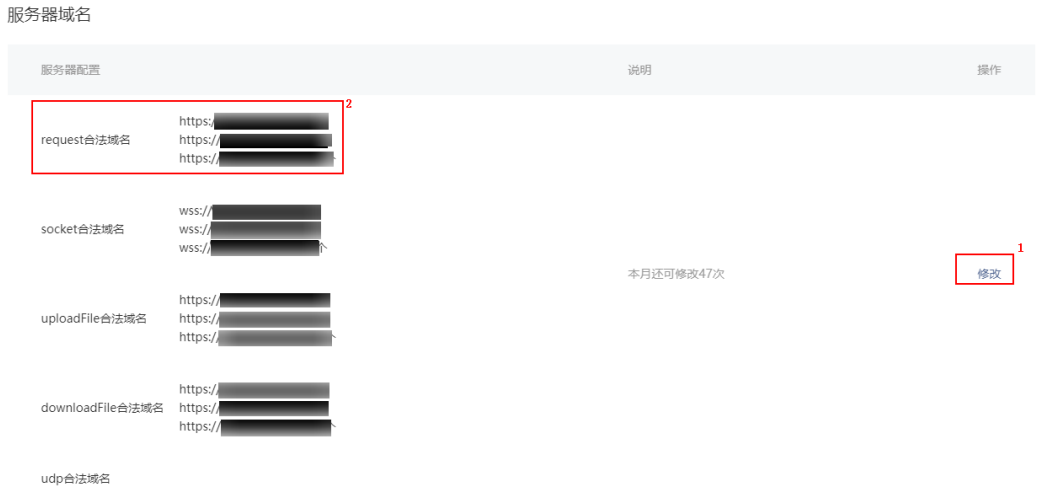


图 10-77 配置业务域名 1

图 10-78 配置业务域名 2

图 10-79 配置“request 合法域名”



- AppID：小程序的AppID，小程序的注册类型为非个人的微信小程序。

  获取方法：使用管理员账号登录**微信公众平台**，在微信公众平台“开发 > 开发设置”中，可查看“AppID（小程序ID）”的值。

- 小程序代码上传密钥：用于上传小程序代码到微信中的密钥。

  获取方法：使用管理员账号登录**微信公众平台**，在微信公众平台“开发 > 开发设置”中，单击“小程序代码上传密钥”后的“重置”，进行获取。

- IP白名单：在AstroZero开发环境发布微信小程序前，需要在微信公众平台“开发 > 开发设置”页面的“小程序代码上传”区域，关闭“IP白名单”。

- 业务域名：使用管理员账号登录**微信公众平台**，在微信公众平台“开发 > 开发设置”页面的“业务域名”区域，单击“修改”。在弹出的“配置业务域名”页面，将承载服务的AstroZero环境访问地址（如华北-北京四AstroZero开发环境域名“appcube.cn-north-4.huaweicloud.com”、运行环境域名为“appcuberun.cn-north-4.huaweicloud.com”），在页面中调用的请求URL（即前端页面中调用的公共接口URL，包括iframe中的请求URL）都加入业务域名中。

  须知

  您需要在“配置业务域名”页面，单击“下载校验文件”，提交工单联系运维人员将校验文件放置在域名根目录下。

- request合法域名：将AstroZero运行环境的域名，配置到微信公众平台的合法域名中。

  配置方法：在微信公众平台“开发 > 开发设置”页面的“服务器域名”区域，单击“修改”。在“request合法域名” 中，添加AstroZero运行环境的域名。AstroZero运行环境是指正式发布上线的真实环境。测试完成后，将应用发布至生产环境，即运行环境，供业务用户使用。华北-北京四AstroZero运行环境域名为“appcuberun.cn-north-4.huaweicloud.com”。

步骤4 使用开发者账号登录AstroZero经典版开发环境，开发免登BO，用于用户通过微信小程序使用发布的应用时，无需用户输入用户名密码，即可获取当前用户身份。

单击**这里**，可获取免登录开发样例。此样例仅作为参考，实际使用时请根据具体场景进行修改，但必须确保入参和出参是一致的。同时需要将服务的请求地址配置到系统参数“token_url_in_wechat”中，且入参、出参设置如下：

- 入参为：
  - code：wx.login后获取的code。
  - appid：当前小程序id。
- 出参为AstroZero用户登录后，生成的token。

说明

同个租户创建不同的应用，发布到微信小程序前，开发免登BO时，为了防止不同小程序的业务用户出现混淆（即为了防止小程序A的业务用户也可登录小程序B），需要对业务逻辑做处理。建议业务用户数据保存到PortalUser表中时，可使用字段“id”代替“usrName”，或者“usrName”前加上应用“id”加以区分。

步骤5 在AstroZero经典版开发环境，创建要发布的小程序应用，在应用内根据需求创建对象、页面、服务编排和脚本等组件。

在AstroZero中如何创建应用，请参见**使用AstroZero创建应用**。

步骤6 登录AstroZero发布小程序。

支持在开发环境、沙箱环境以及运行环境中发布小程序，小程序中调用的后台接口服务所承载的环境，取决于在哪个环境中发布小程序。

- 在经典版开发环境中，发布小程序步骤如下：
  a. 在经典版应用开发页面右下角，单击  ，选择“编译”。
  b. 编译成功后，页面会显示“编译*XXX*成功”，关闭编译成功的窗口。
  c. 在应用开发页面左侧，单击  ，选择“微信”。
     在“发布微信小程序”页面，配置如下参数。
     - 小程序ID：小程序的AppID，参数获取方法请参见**步骤3**。
     - 版本号：发布的版本号。每次发布时需要使用新的版本号，使用之前请确认版本号是否已被使用过。
     - 主页：选择一个页面作为首页，即显示的默认页面。
     - 小程序代码上传密钥：用于上传小程序代码到微信中的密钥，参数获取方法请参见**步骤3**。
  d. （可选）单击“预览”，可生成小程序预览页面的二维码，开发者使用微信扫一扫功能，可预览小程序页面。
  e. 设置完成后，单击“发布”。
- 在沙箱环境及经典版运行环境中，发布小程序步骤如下（需要提前在沙箱环境或经典版运行环境中，安装业务应用，具体操作可参考**如何安装应用**）：
  a. 在经典版运行环境管理中心，选择“应用管理 > 应用导航”。
  b. 在应用程序列表中，单击待发布应用后的  ，再选择  。
  c. 在“发布微信小程序”页面，配置如下参数。
     - 小程序ID：小程序的AppID，参数获取方法请参见**步骤3**。
     - 版本号：发布的版本号。每次发布时需要使用新的版本号，使用之前请确认版本号是否已被使用过。
     - 主页：选择一个页面作为首页，即显示的默认页面。
     - 小程序代码上传密钥：用于上传小程序代码到微信中的密钥，参数获取方法请参见**步骤3**。
  d. （可选）单击“预览”，可生成小程序预览页面的二维码，开发者使用微信扫一扫功能，可预览小程序页面。
  e. 设置完成后，单击“发布”。

步骤7 使用小程序开发者账号登录**微信公众平台**，选择“管理 > 版本管理”，在上传的小程序后单击“提交审核”。

您也可以在“版本管理”页面，单击小程序后的下拉图标，将小程序设置为体验版，生成小程序预览页面的二维码，体验者用户可通过微信扫一扫功能预览小程序。

图 10-80 上传小程序

审核通过后（审核时长一般为一周内），在微信小程序中可搜索到上传的应用。

----结束

11 业务用户专项

11.1 通过 AstroZero 开发业务用户登录页

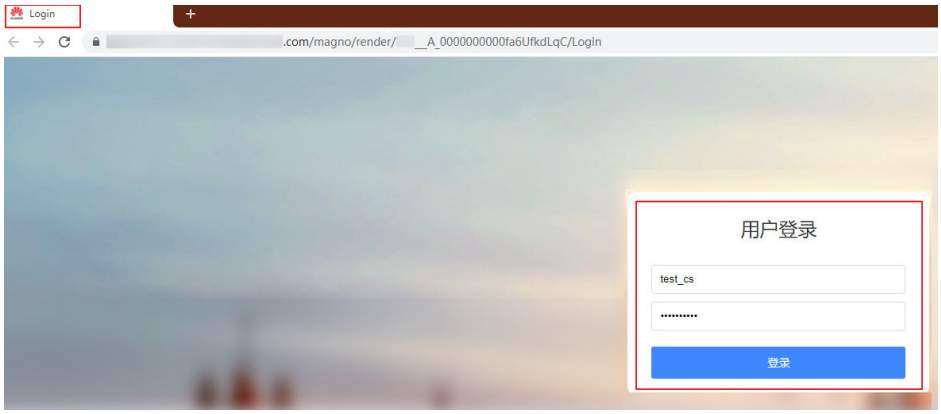
11.1.1 业务用户登录页方案概述

应用场景

使用AstroZero开发完应用后，系统会为应用预置一个默认的登录页，业务用户可通过默认的登录页登录应用。除了使用默认的登录页，您还可以根据自身业务的实际需求，使用AstroZero为应用自定义一个登录页。通过在登录页中输入用户名、密码和手机号码等登录信息，与系统中存储的业务用户信息进行对比，来验证业务用户的身份，并根据设置的业务用户权限，为业务用户分配相应的资源和访问权限。

例如，自定义一个**图11-1**中的登录页，在登录页中输入业务用户名及密码，单击“登录”，即可登录应用。其中，“登录”逻辑是通过“自定义登录”组件，调用用户登录服务编排完成的。在自定义登录页前，请先了解下方的**业务用户的登录方式**和**业务用户的登录机制**。

图 11-1 自定义登录页面



业务用户的登录方式

业务用户登录AstroZero有两种登录方式：后台登录和前台登录。

- 业务用户在后台登录时，是使用自定义的服务编排来调用“login”脚本，查询登录账号密码，判断当前登录的账号密码是否正确，来实现业务用户后台登录功能。
- 业务用户在前台登录时，需要先在线下开发一个登录组件，上传到高级页面，并在高级页面中配置组件桥接器中的数据。最后在页面中输入登录账号密码，通过调用“用户登录服务编排”，实现业务用户页面登录功能。

业务用户的登录机制

业务用户前台登录和后台登录，在登录过程中的服务逻辑实现过程如下：

1. 通过调用“账号密码校验”脚本，查询登录账号密码，判断当前登录的账号密码是否正确。
2. 如果判断账号密码错误，直接执行“账号密码错误”。账号密码正确，继续判断是否有验证码。
3. 如果判断当前登录没有验证码，则直接执行登录。当前有验证码，则继续判断验证码是否正确。
4. 如果判断验证码正确，则执行登录操作，验证码错误，则执行验证失败。

11.1.2 业务用户登录页后端逻辑开发实施步骤

业务用户在后台登录时，是使用自定义的服务编排，来调用“账号密码校验”脚本，查询登录账号密码，判断当前登录的账号密码是否正确来实现“业务用户”后台登录功能的。“业务用户登录”服务编排开发的大致过程为：先拖拽1个脚本图元，3个决策图元以及3个赋值图元，再分别配置各个图元属性，然后配置各个图元之间连线类型，最后保存启用。

下面以“A”应用为例，介绍如何在后台实现业务用户登录。

步骤一：创建密码校验登录脚本

业务用户登录前，需要先创建一个账号密码校验脚本。

步骤1 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。

步骤2 在主页中，单击“进入首页”，进入AstroZero应用开发页面。

步骤3 在“主页 > 全部应用”中，单击“A”应用后的“编辑”，进入应用设计器。

步骤4 在左侧导航栏中，选择“逻辑”，单击脚本后的“+”。

步骤5 选中“创建一个新脚本”，“名称”设置为“login”，单击“添加”。

图 11-2 新增脚本

步骤6 在代码编辑器中，插入如下脚本代码。

```
import * as buffer from"buffer";
import * as crypto from"crypto";
import * as db from"db";
//定义入参结构，账号的用户名、密码为必需字段。如果根据业务需要，需校验其他字段（例如验证码），则根据账号密码字段的格式进行新增即可。
@action.object({type:"param"})
export class ActionInput{
    @action.param({type:'String', required:true, label:'string'})
    username:string;    //用户名
    @action.param({type:'String', required:true, label:'string'})
    password:string;    //密码
    @action.param({type:'String', required:true, label:'string'})
    captcha:string;     //验证码，本脚本只是为了校验账号密码，因此用不到验证码，验证码也不是必需字段。在实现业务用户后台登录的Flow中，Flow调用此脚本，需要判断验证码，所以在此脚本中添加了验证码字段。
}
//定义出参结构，结构中的字段可以根据业务需要按下方样例结构进行添加或减少。
@action.object({type:"param"})
export class ActionOutput{
    @action.param({type:'String'})
    msg:string;//登录信息
    @action.param({type:'String'})
    username:string;//用户名
    @action.param({type:'String'})
    userId:string;//用户ID
    @action.param({type:'String'})
    captcha:string;//验证码
}
//使用数据对象PortalUser
@useObject(['PortalUser'])
@action.object({type:"method"})
export class Login{
//定义接口类，接口的入参为ActionInput，出参为ActionOutput
    @action.method({ input:'ActionInput', output:'ActionOutput'})
    public login(input:ActionInput):ActionOutput{
        let out =new ActionOutput();
        //新建出参ActionOutput类型的实例，作为返回值
        let error =new Error();
```

```
    //新建错误类型的实例，用于在发生错误时保存错误信息
    try{
        out.captcha = input.captcha;
        let s = db.object('PortalUser');
        let condition ={
        "conjunction":"AND",
        "conditions":[{
            "field":"usrName",
            "operator":"eq",
            "value": input.username
            }]
        };
        let user = s.queryByCondition(condition);
        if(user && user.length ==1){
            if(validate(user[0].passwordSalt, user[0].userPassword, input.password)){
                out.msg ="登录成功！";
                out.username = user[0].usrName;
                out.userId = user[0].id;
            }else{
                out.msg ="账号或者密码错误！";
            }
        }else{
            out.msg ="账号或者密码错误！";
        }
    }catch(error){
        console.error(error.name, error.message);
        out.msg = error.message;
    }
    return out;
  }
}
function _salt(password:string, saltBuf: buffer.Buffer, encoding: buffer.Encoding=
buffer.Encoding.Base64):string{
  const passwordBuf = buffer.from(password)
  const crypt = crypto.pbkdf2(passwordBuf, saltBuf,1000,32, crypto.Hashs.SHA1)
  return crypt.toString(encoding)
}
function validate(salt:string, userSaltedPassword:string, password:string, encoding: buffer.Encoding=
buffer.Encoding.Base64):boolean{
  const saltBuf = buffer.from(salt, encoding);
  const saltedPassword = _salt(password, saltBuf, encoding);
  return saltedPassword === userSaltedPassword
}
```

步骤7 单击编辑器上方的  ，保存脚本。

步骤8 测试脚本能否正常执行。

1. 单击编辑器上方的  ，执行脚本。
2. 在界面底部的输入参数中，输入如下测试数据，单击  。
   其中，“test_cs”、“*{XXXXXXXX}*”为注册的业务用户账号和密码，“captcha”验证码非必填项为空。

   说明

   在业务配置中心创建的业务用户和使用脚本创建的业务用户不能通用，故此处设置的业务用户（如test_cs）不能为在业务配置中心创建的业务用户。业务配置中心创建的业务用户，只能在默认登录页中使用。

   ```
   {
     "username": "test_cs",
     "password": "{XXXXXXXX}",
     "captcha": ""
   }
   ```

   执行成功后，在“输出参数”中可参看结果。

图 11-3 返回结果

步骤9 测试成功后，单击编辑器上方的，启用该脚本。

----结束

步骤二：通过服务编排开发登录页后端逻辑

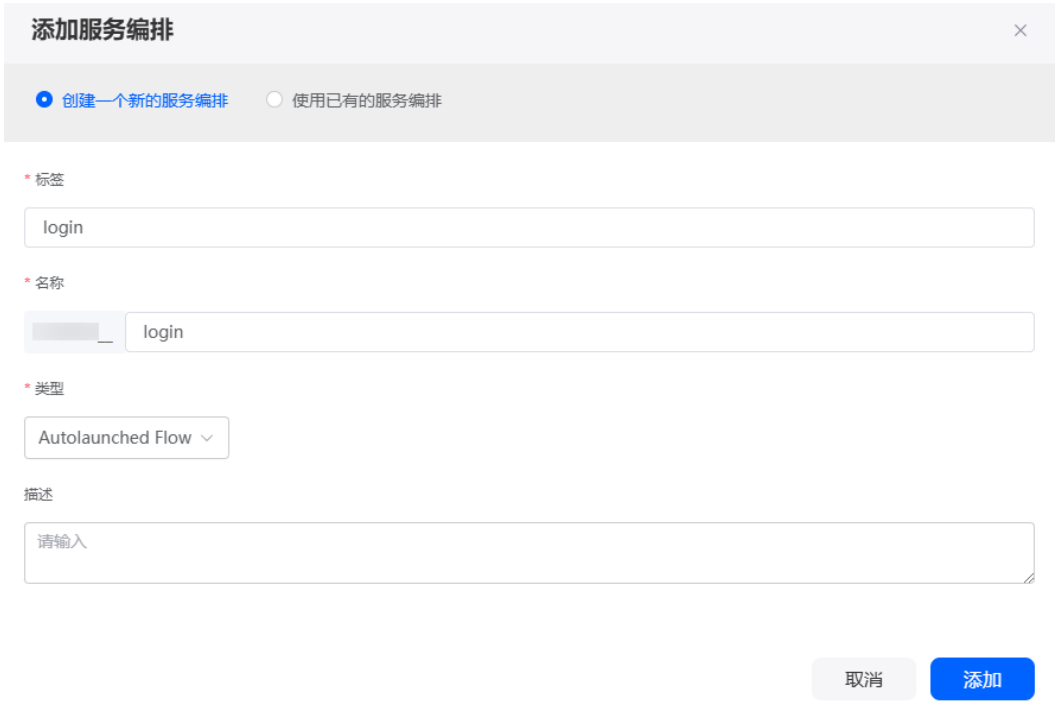
步骤1 在“A”应用的设计器中，单击左侧导航栏的“逻辑”，单击编排后的，新建一个Flow目录。

图 11-4 创建目录

步骤2 将鼠标放在“Flow”上，单击界面上出现的“+”，进入添加服务编排页面。

步骤3 选中“创建一个新的服务编排”，“标签”和“名称”设置为“login”，类型设置为“Autolaunched Flow”，单击“添加”。

图 11-5 创建服务编排



步骤4 定义服务编排用到的变量。

1. 单击，展开全局上下文，再单击“变量”后的，设置参数名称为“username”。

图 11-6 新增变量

2. 重复上一步，定义表11-1中其他变量。

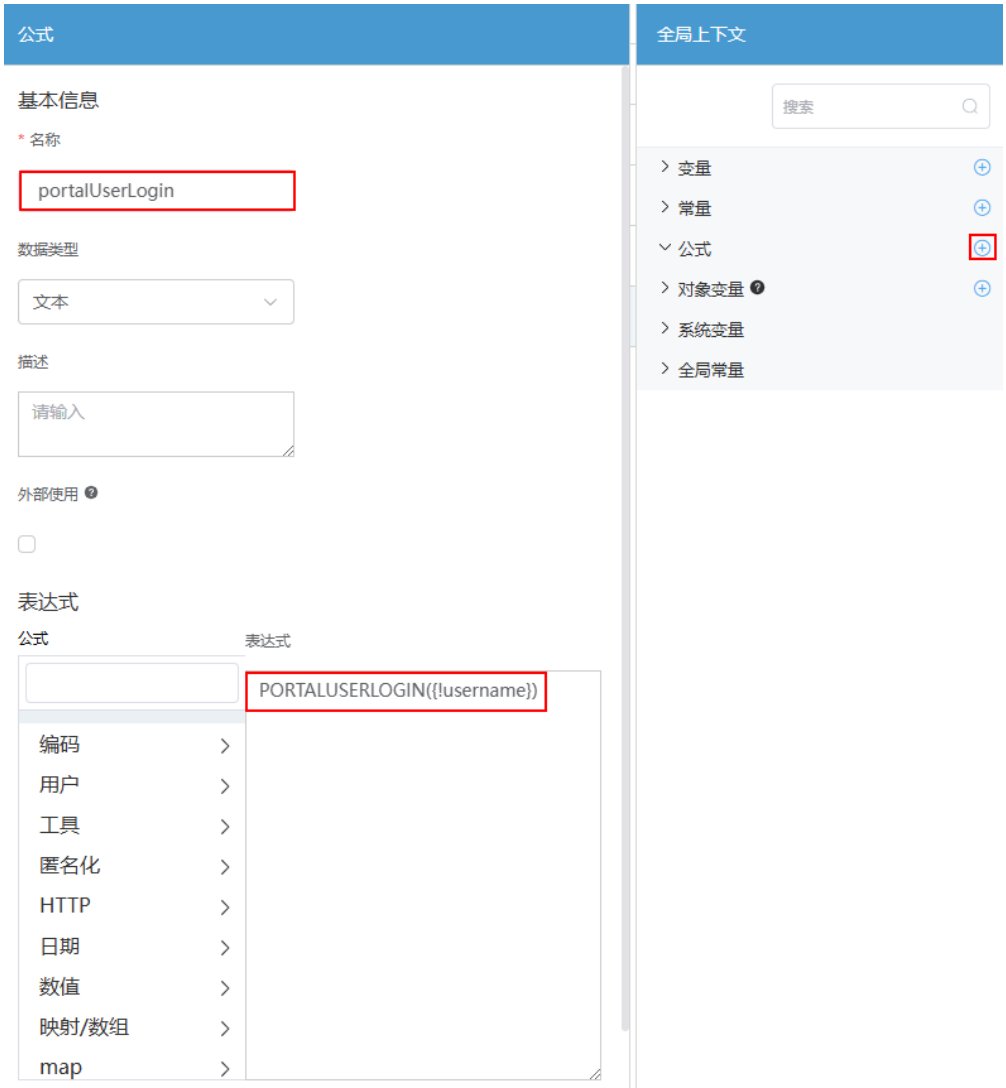
表 11-1 服务编排变量说明

名称（变量名称，唯一标识）	数据类型
username	文本
password	文本

名称（变量名称，唯一标识）	数据类型
captcha	文本
msg	文本
userId	文本
loginMsg	文本

3. 单击“公式”后的⊕，在左侧公式弹窗中，设置“名称”为“portalUserLogin”，“表达式”为“PORTALUSERLOGIN({!username})”，单击“保存”。

图 11-7 添加公式变量“portalUserLogin”



4. 参考上一步，创建表11-2中公式变量“verifyCode”。

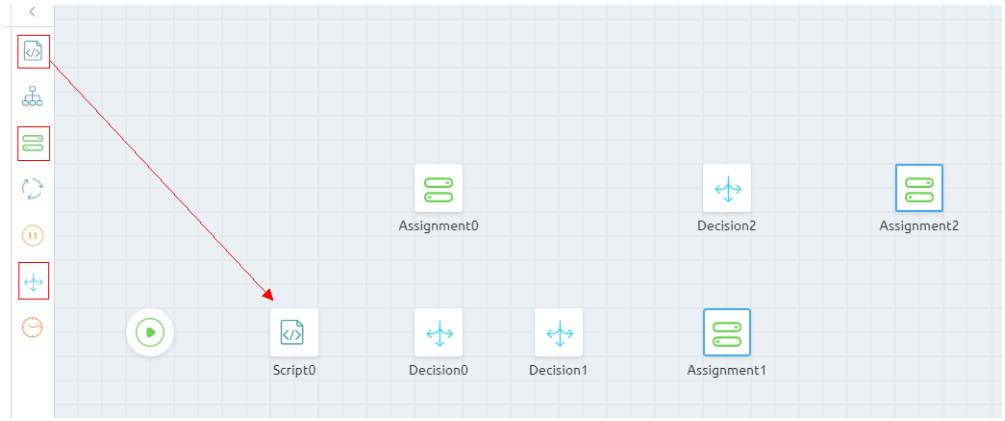
表 11-2 公式变量说明

名称	表达式
portalUserLogin	PORTALUSERLOGIN({!username})
verifyCode	VERIFYCODEWITHTYPE({!captcha},"login")

步骤5 拖拽图元到服务编排画布，并配置图元的基本属性。

1. 从图元区分别拖拽脚本（1个）、决策（3个）、赋值（3个）图元到画布中，图元排列如下图所示。

图 11-8 图元排列



2. 选中“Script0”图元，在右侧基本信息中，设置“标签”为“查询用户”。
3. 参考上一步，设置其他图元的“标签”属性，具体值如下表所示。

表 11-3 设置其他图元标签属性

名称（变量唯一标识，不需要修改）	标签
Decision0	判断账号密码
Decision1	判断是否包含验证码
Decision2	校验验证码
Assignment0	账号密码错误
Assignment1	执行登录
Assignment2	验证失败

图 11-9 修改后图元

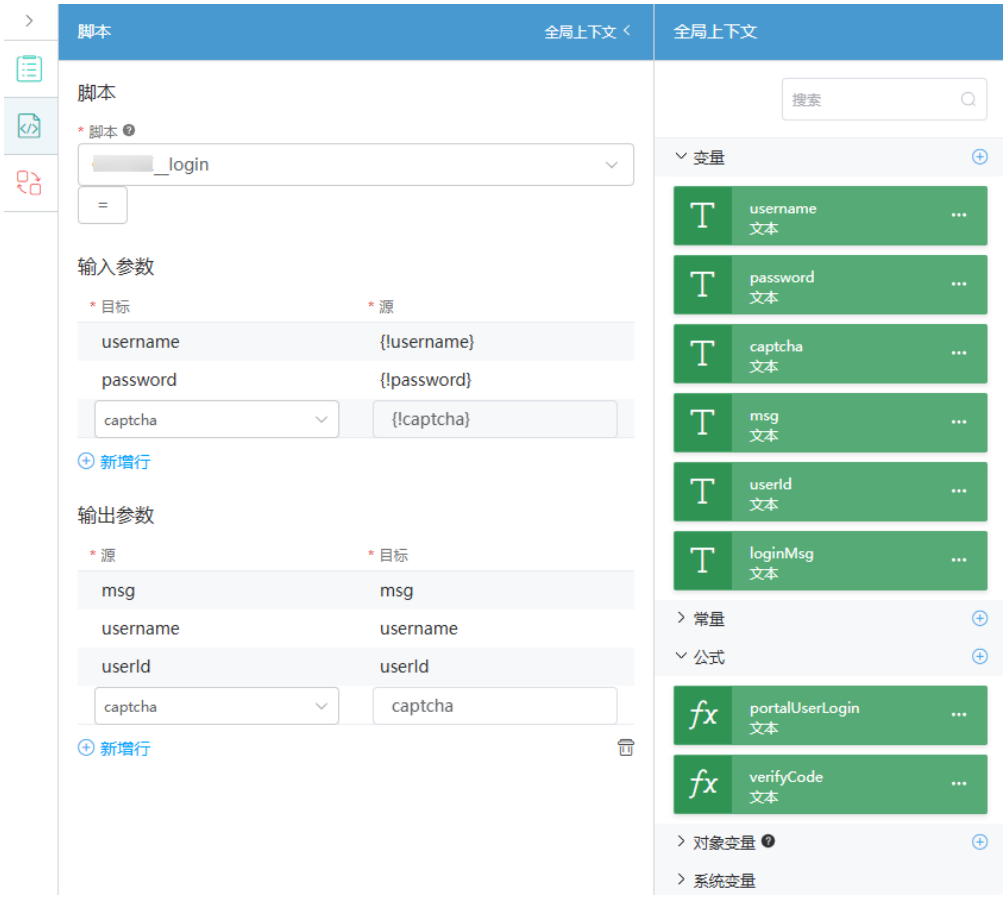
步骤6 配置“查询用户”脚本图元。

1. 单击，指定图元对应的脚本名称（*命名空间*__login），并配置脚本的输入输出参数。

图 11-10 指定脚本

2. 单击“全局上下文”，显示变量列表，从“变量”中，拖拽“username”、“password”和“captcha”到“输入参数”下对应的“源”输入框中，在“输出参数”下，单击4次“新增行”，依次添加下拉选项中的输出参数字段，并从“变量”中拖拽相应的字段到“目标”输入框下，字段与变量对应关系如下图所示。
   - 脚本图元中，输入参数、输出个数和指定脚本中需要的输入参数字段数是一致的。如果自定义脚本的输入参数有额外字段，额外的字段也需要同样操作。
   - 请直接从全局上下文拖拽“变量”到对应的输入输出参数下，如果手动输入请确认输入参数与全局上下文中变量的值一致。

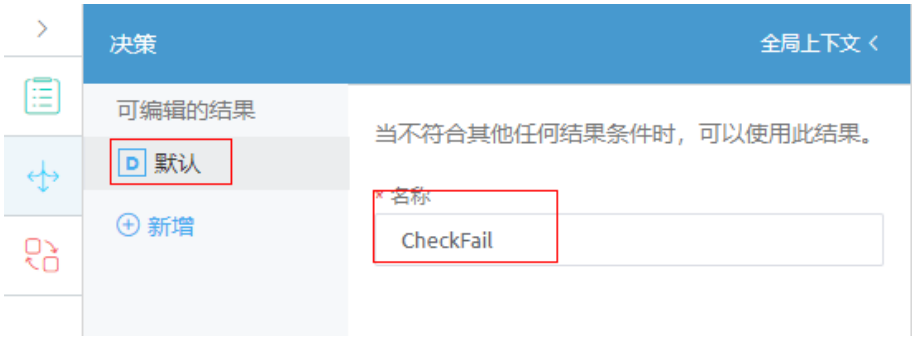
图 11-11 拖拽脚本的输入输出参数



步骤7 配置“判断账号密码”决策图元。

1. 选择“判断账号密码”图元，在右侧单击图标，修改“默认”的“名称”为“CheckFail”。

图 11-12 修改“默认”结果名称



2. 单击“新增”，增加一个可编辑的结果，修改结果为“CheckSuccess”，在“可视”下单击“新增行”，并拖拽变量中的“msg”到“资源”下，设置“比较符”为“==”，“值”为“"登录成功！"”。

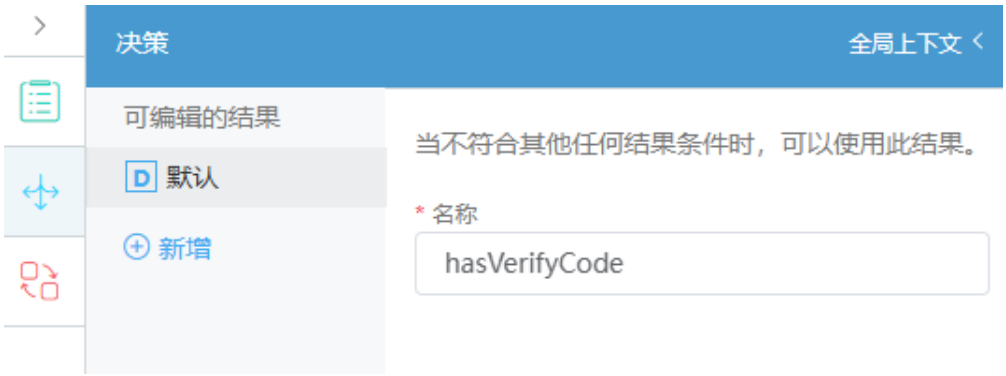
图 11-13 修改可编辑的结果

- 请直接从全局上下文拖拽变量“msg”到“资源”下，如果手动输入请确认输入参数与全局上下文中变量的值一致。
- “登录成功！”需要与“login”登录脚本中的输出参数一致。

步骤8 配置“判断是否包含验证码”决策图元。

1. 选择“判断是否包含验证码”图元，在右侧单击图标，修改“默认”的“名称”为“hasVerifyCode”。

图 11-14 修改默认结果名称



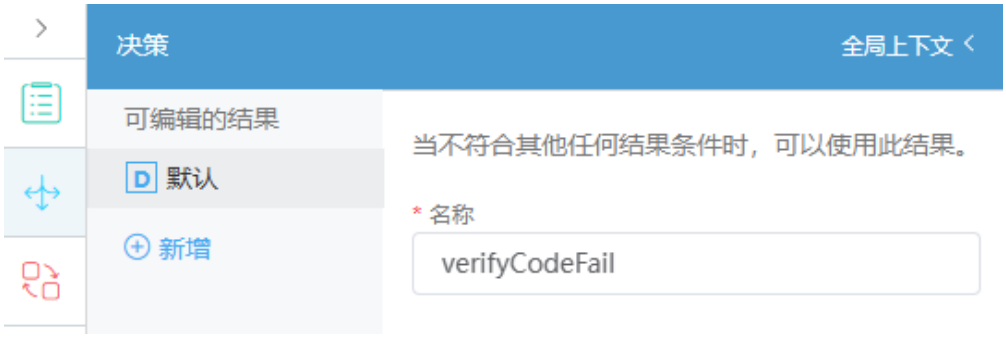
2. 单击“新增”，增加一个可编辑的结果，修改结果为“noVerifyCode”，在“可视”下单击“新增行”，并拖拽变量中的“captcha”到“资源”下，设置“比较符”为“==”，“值”为“""”。

图 11-15 修改可编辑的结果

步骤9 配置“校验验证码”决策图元。

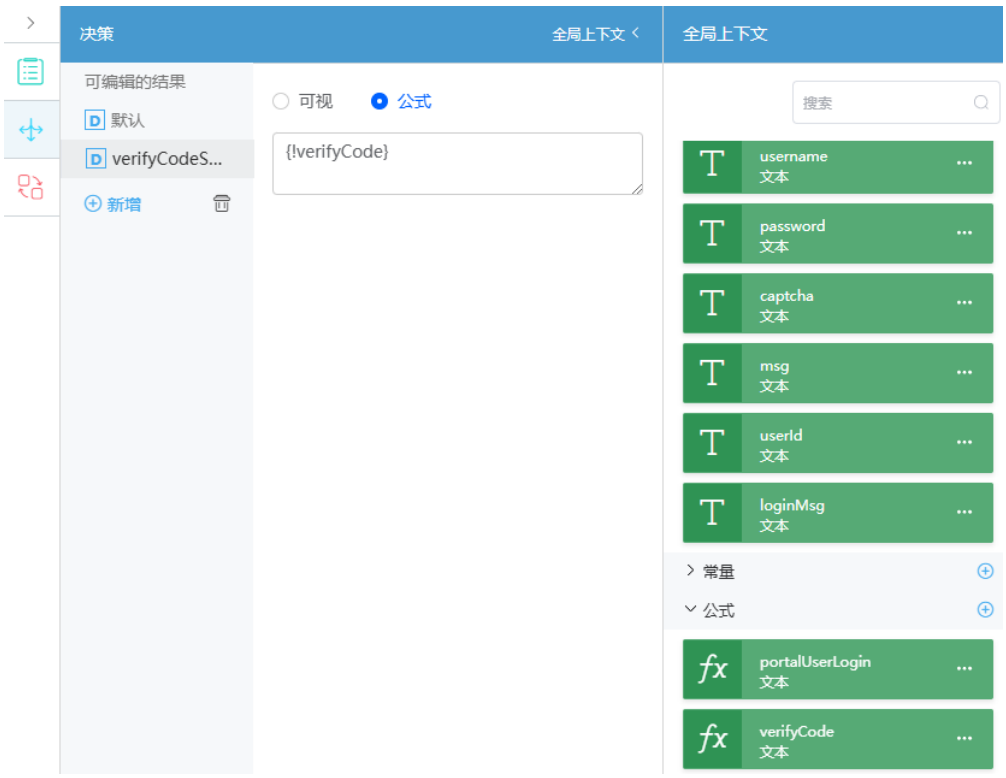
1. 选择“校验验证码”图元，在右侧单击图标，修改“默认”的“名称”为“verifyCodeFail”。

图 11-16 修改“默认”名称



2. 单击“新增”，增加一个可编辑的结果，修改结果为“verifyCodeSuccess”，在右侧选择“公式”，并从全局上下文中，拖拽“verifyCode”到“公式”下。

图 11-17 修改可编辑的结果



步骤10 配置“账号密码错误”赋值图元。

图 11-18 配置“账号密码错误”图元

1. 选择“账号密码错误”图元，在右侧单击图标，单击“新增行”。

2. 从全局上下文的“系统变量”中，拖拽“$Flow.ResMsg”到“赋值”下，并设置“操作符”为“=”，拖拽“msg”到“值”。
3. 单击“新增行”，拖拽“系统变量”下的“$Flow.ResCode”到“赋值”的“变量”下，设置“操作符”为“=”，设置“值”为“"1"”。

步骤11 配置“执行登录”赋值图元。

图 11-19 拖拽“执行登录”赋值的变量及值

1. 选择“执行登录”图元，在右侧单击图标，单击4次“新增行”。
2. 从全局上下文，拖拽“msg”等字段到“赋值”的“变量”下，并设置“操作符”为“=”，然后再拖拽“值”下的各个值，具体字段对应关系，如下图所示。

表 11-4 变量与值对应关系

变量	操作符	值
loginMsg	=	portalUserLogin
msg	=	msg
username	=	username
userId	=	userId

步骤12 配置“验证失败”赋值图元。

图 11-20 配置“验证失败”赋值图元

1. 选择“验证失败”图元，在右侧单击图标，单击“新增行”。
2. 从全局上下文“系统变量”，拖拽“$Flow.ResMsg”、“$Flow.ResCode”到“赋值”下，并设置操作符为“=”，分别设置“值”为“"账号或者密码错误！"”、“"1"”。

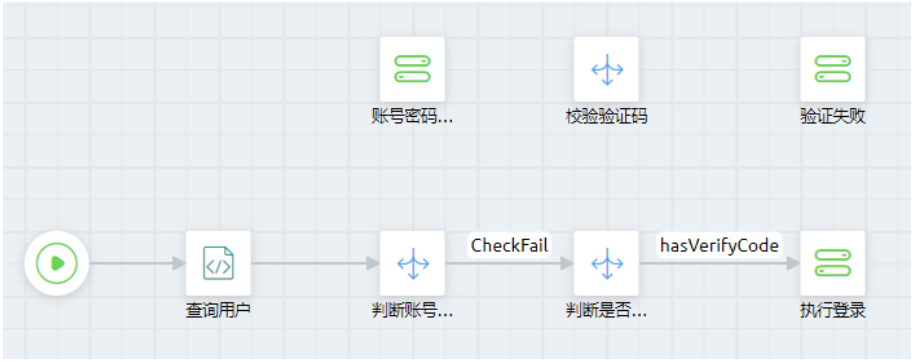
表 11-5 赋值

变量	操作符	值
$Flow.ResMsg	=	"账号或者密码错误！"
$Flow.ResCode	=	"1"

步骤13 拖拽图元连线，并配置连线属性。

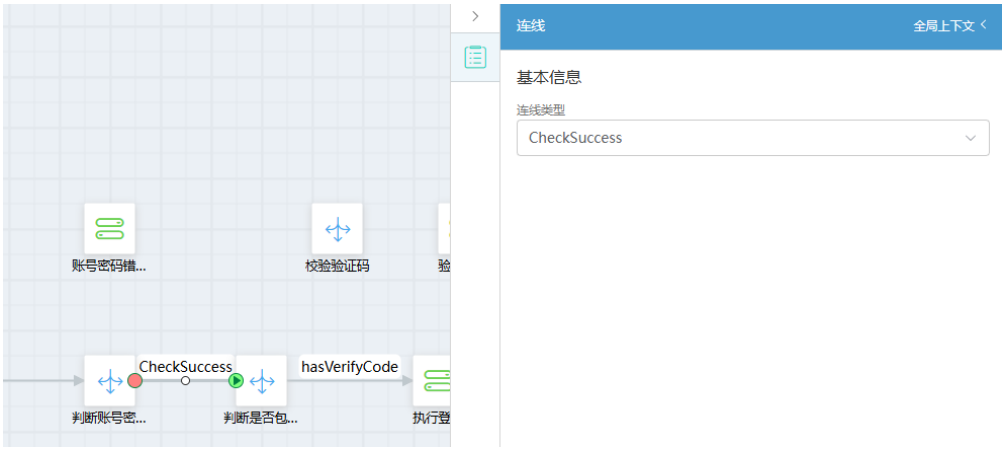
1. 在画布上，把鼠标放在起点图元上，从“+”拖动鼠标，在起点图元和“查询用户”图元间增加连线；即将当前脚本设置为服务编排的起始节点。
2. 依次在“查询用户”、“判断账号密码”、“判断是否包含验证码”、“执行登录”图元直接拖拽连线。

图 11-21 拖拽连线



3. 单击“判断账号密码”与“判断是否包含验证码”图元之间的连线，在右侧属性单击，在“连线”中修改“连线类型”为“CheckSuccess”。

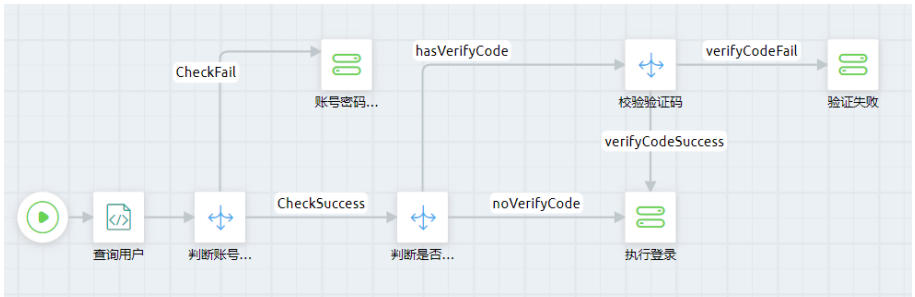
图 11-22 选中连线



4. 单击“判断是否包含验证码”与“执行登录”图元之间的连线，在右侧属性单击，在“连线”中修改“连线类型”为“noVerifyCode”。
5. 从“判断账号密码”图元上，拖拽一条连线到“账号密码错误”图元。
6. 从“判断是否包含验证码”图元上，拖拽一条连线到“校验验证码”图元。
7. 从“校验验证码”图元上，拖拽一条连线到“验证失败”图元。
8. 从“校验验证码”图元上，拖拽一条连线到“执行登录”图元，并设置该连线的“连线类型”为“verifyCodeSuccess”。

连线拖拽完成，如下图所示。

图 11-23 拖拽图元连线



步骤14 定义服务编排的输入、输出参数，并保存服务编排。

1. 鼠标在画布空白处点一下，单击右侧，设置服务编排的输入输出参数，如下图所示。

图 11-24 拖拽服务编排的输入输出参数

- 服务编排的输入参数是用来执行服务编排时输入的参数，同时也是执行账号密码校验脚本时的输入参数。所以当账号密码校验脚本的有额外的输入参数字段，服务编排的输入参数也需要同步增加。
- 服务编排的输出参数是执行账号密码校验脚本时返回的参数，所以当账号密码校验脚本的有额外的输出参数字段，服务编排的输出参数也需要同步增加。

2. 单击服务编排页面上方的，保存服务编排。

步骤15 测试服务编排能否正常执行。

1. 单击服务编排页面上方的，进入服务编排测试页面。

图 11-25 服务编排测试页面

2. 在“Flow Run”界面中，输入测试数据，单击“运行”。
   其中，“test_cs”、“***”为业务用户的账号和密码。

```
{
 "username": "test_cs",
 "password": "***",
 "captcha": ""
 }
```

   执行成功后，界面上会返回设备对象中的全部信息，示例如下：

图 11-26 返回值示例

返回值提示登录成功，完成业务用户的登录。业务用户登录成功后，返回AstroZero，刷新页面后在页面右上角可以看到当前登录的用户已变成在服务编排中输入的业务用户。

步骤16 测试成功后，单击编辑器上方的  ，启用并发布服务编排。

----结束

步骤三：新建开放接口

开放接口是用户将在应用中开发的脚本、服务编排等包装成自定义REST接口的形式，供其他系统进行调用。本示例将开发的服务编排包装发布REST接口，供**业务用户登录页前台开发实施步骤**中操作调用。

步骤1 在“A”应用的设计器中，单击左侧导航栏的“集成”。

步骤2 单击开放接口后的“+”，进入新建开放接口页面。

步骤3 设置接口参数，单击“保存”。

图 11-27 新建开放接口

表 11-6 开放接口参数说明

参数	说明	示例
标签	新建API接口的标签，长度不能超过64个字符。	login
操作名称	新建API接口的操作名称，必须以英文字母开头，由英文字母，数字和单下划线组成，不能以单下划线结尾，且长度不能超过40个字符。	login
版本	API接口的版本，格式为“x.y.z”。	1.3.3
URL	API接口路径， 固定以 /service/*{命名空间}__{应用名}/{版本}* 开头， 后面接API的具体路径。此处设置的内容是新建开放接口提供给外部访问的URL。	/service/*命名空间*__A/*版本*/Flow_login
类型	选择资源类型，只有服务编排类型的接口能够在服务编排中被调用，其他类型接口只能通过API的形式进行调用。	服务编排

参数	说明	示例
资源	根据类型选择需要绑定的资源。	选择**步骤二：通过服务编排开发登录页后端逻辑**中创建的服务编排，请确保服务编排已启用，否则此处选择不到。
方法	API接口的HTTP方法。 本示例选择“POST”，即请求服务器新增资源或执行特殊操作。	POST

----结束

11.1.3 业务用户登录页前台开发实施步骤

业务用户在前台登录时，需要先在线下开发一个登录组件，上传到高级页面，并在高级页面中配置组件桥接器中的数据。最后在页面中，输入登录账号密码，通过调用“用户登录服务编排”，实现“业务用户”页面登录功能。

本文以“A”应用为例，介绍如何开发登录页面，具体流程如**图11-28**所示。

图 11-28 应用登录页面开发流程

步骤一：开发自定义组件

本示例中，提供了一个已开发好的自定义登录组件userLogin，单击**userLogin.zip**可直接下载使用。如果想要了解自定义登录组件的开发方法，可参考本步骤执行。示例中的userLogin.zip包为vue2架构的，使用前请参考**关闭Vue3框架渲染组件开关**中操作，将页面组件的渲染框架切换回Vue2。

步骤1 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。

步骤2 单击，选择“环境管理 > 环境配置”，进入环境配置。

步骤3 在主菜单中，选择“维护”。

步骤4 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 组件模板”。

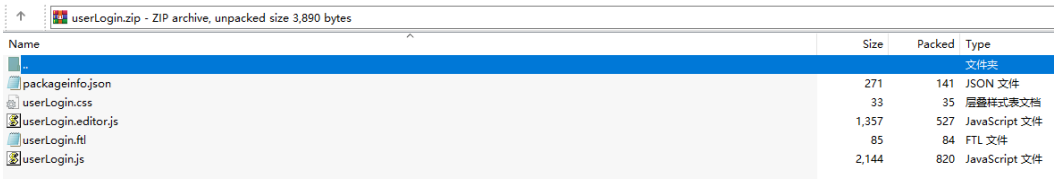
步骤5 在组件模板列表中，单击widgetVueTemplate，进入模板详情页。

步骤6 单击“下载”，设置组件的名称为“userLogin”，单击“保存”，将模板下载到本地。

图 11-29 保存模板

步骤7 查看解压后的组件目录。

图 11-30 解压后目录



- userLogin.js：存放vue业务逻辑的代码，请根据业务需求自行开发。
- userLogin.ftl：存放html代码，请根据业务需求自行开发。
- userLogin.css：存放样式代码，请根据业务需求自行开发。
- userLogin.editor.js、packageinfo.json：配置文件，请参考**步骤8**和**步骤9**修改。

步骤8 修改userLogin.editor.js文件中的config代码，用于配置桥接器。

代码示例如下：

```
config: [
        {
            type: 'connectorV2',
            name: 'FlowConnector',
            label: 'Flow Connector',
            model: 'ViewModel',
        },
        {
            type: 'connectorV2',
            name: 'common.GetConnector',
            label: 'View API Get Connector',
            model: 'ViewModel',
        },
        {
            type: 'connectorV2',
            name: 'common.PostConnector',
            label: 'View API Post Connector',
            model: 'ViewModel',
        },
        {
            type: 'connectorV2',
            name: 'common.PutConnector',
            label: 'View API Put Connector',
            model: 'ViewModel',
        },
        {
```



```
                type: 'connectorV2',
                name: 'common.DeleteConnector',
                label: 'View API Delete Connector',
                model: 'ViewModel',
            },
        ]
```

步骤9 修改packageinfo.json文件中的“requires”。

代码示例如下加粗内容：

```
{
  "widgetApi": [
    {
      "name": "userLogin"
    }
  ],
  "widgetDescription": "",
  "authorName": "",
  "localFileBasePath": "",
  "requires": [
    {
      "name": "global_Vue3",
      "version": "3.4.21"
    },
    {
      "name": "global_ElementPlus",
      "version": "2.6.0"
    },
    {
      "name": "global_Vue3I18n",
      "version": "9.10.1"
    },
    {
      "name": "global_Vue3Router",
      "version": "4.3.0"
    }
  ]
}
```

步骤10 将修改后的配置文件和自定义开发的文件，压缩成一个zip包。

----结束

步骤二：上传自定义登录组件

自定义组件开发完成后，需要上传到AstroZero组件库中，供高级页面使用。

步骤1 在AstroZero环境配置页面，单击主菜单中的“维护”。

步骤2 在左侧导航栏中，选择“全局元素 > 页面资产管理 > 组件”。

步骤3 单击“提交新组件”，进入提交新组件页面。

步骤4 单击“上传”，上传自定义组件包“userLogin.zip”。

图 11-31 上传自定义组件

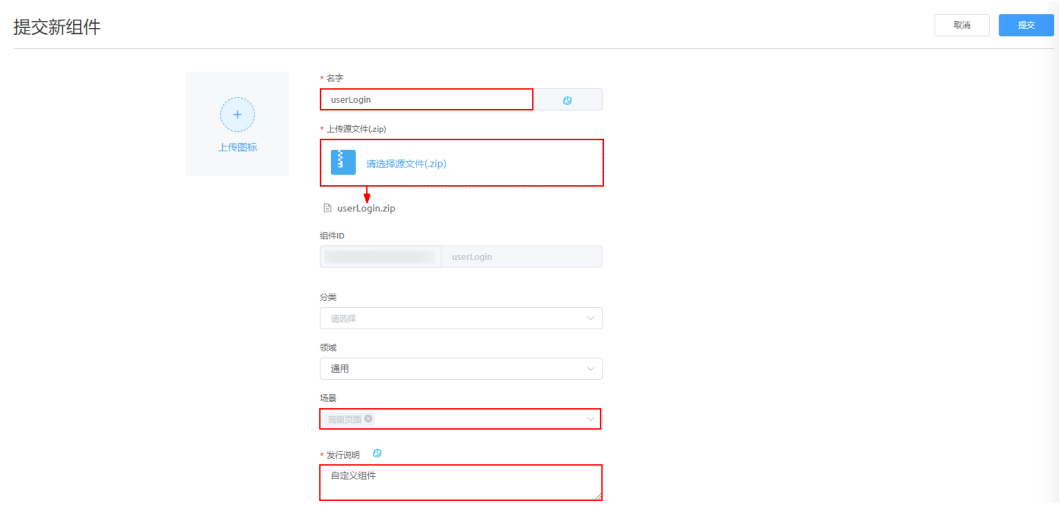


表 11-7 上传组件参数说明

参数	说明	示例
名字	新提交组件的名称，系统会根据组件包名称自动填充。	userLogin
上传源文件	组件的源文件包。	选择**步骤一：开发自定义组件**中的自定义组件登录包
场景	选择组件包的应用场景，可同时勾选多项，勾选后，在相应类型页面开发中，才可使用该组件。	高级页面
发行说明	组件的描述信息，需要配置不同语种下的描述信息。 此处配置的信息，将会在组件详情页的“概况”页签中进行展示。	自定义组件

步骤5 参数设置完成后，单击“提交”，完成组件的上传。

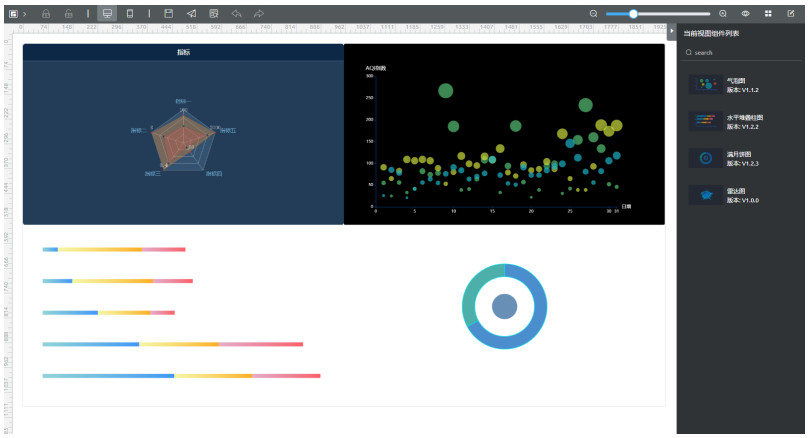
----结束

步骤三：创建高级页面

“业务用户登录”页面是一个高级页面，主要是通过引用上传的自定义登录组件，再配置相关参数，来实现登录功能。

在组装业务用户登录页面时，需要配置从登录页面登录成功后跳转的页面，所以需要提前创建一个高级页面。在本示例中，假设已创建了一个Home页面。Home页面中没有实际数据，在Home页面中拖拽了几个图表组件，然后保存发布成样例页面。在实际开发过程中，可以根据业务需要，选择跳转到需要的页面，或者执行自定义的业务逻辑。

图 11-32 Home 页面



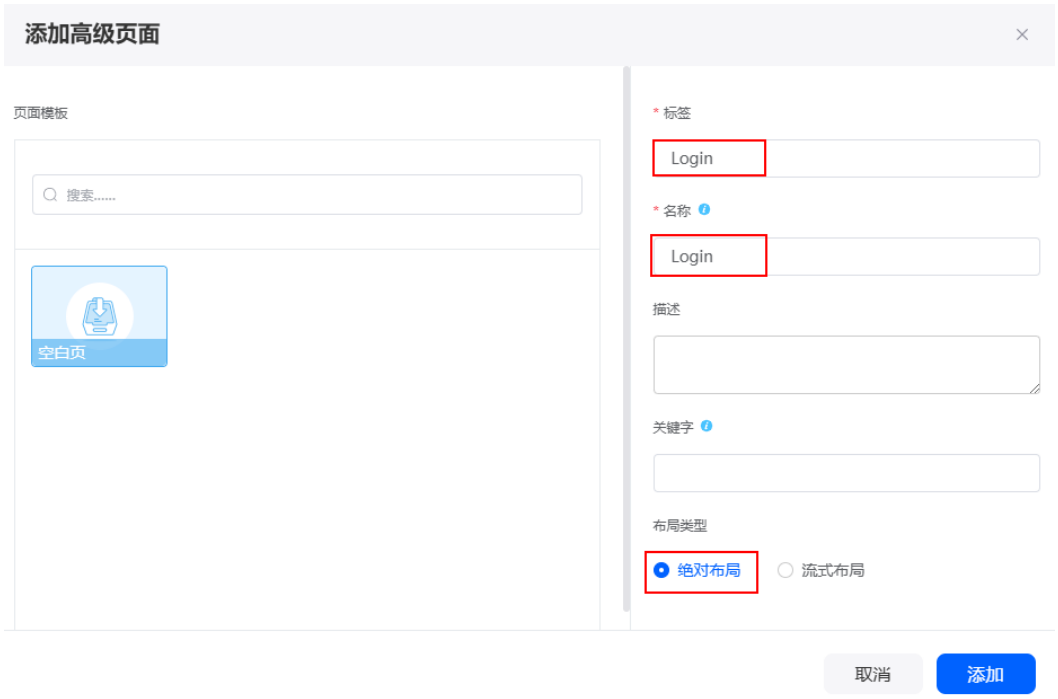
步骤1 在AstroZero服务控制台的主页中，单击“进入首页”，进入AstroZero应用开发页面。

步骤2 在“主页 > 全部应用”中，单击应用“A”后的编辑，进入应用“A”的设计器。

步骤3 在左侧导航栏中，选择“界面”。

步骤4 单击高级页面后的“+”，设置标签和名称为“Login”，并选择“绝对布局”，单击“添加”。

图 11-33 添加高级页面

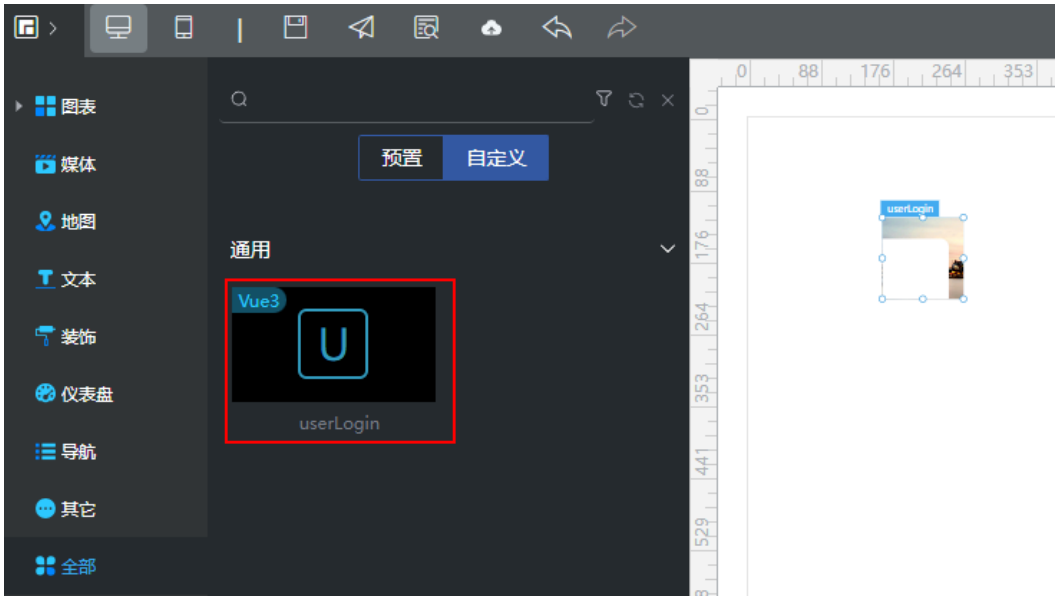


----结束

步骤四：拖拽组件开发登录页面

步骤1 在**步骤三：创建高级页面**中创建的高级页面中，单击左上角的，选择“全部”，在“自定义”页签，拖拽“userLogin”组件到页面编辑区。

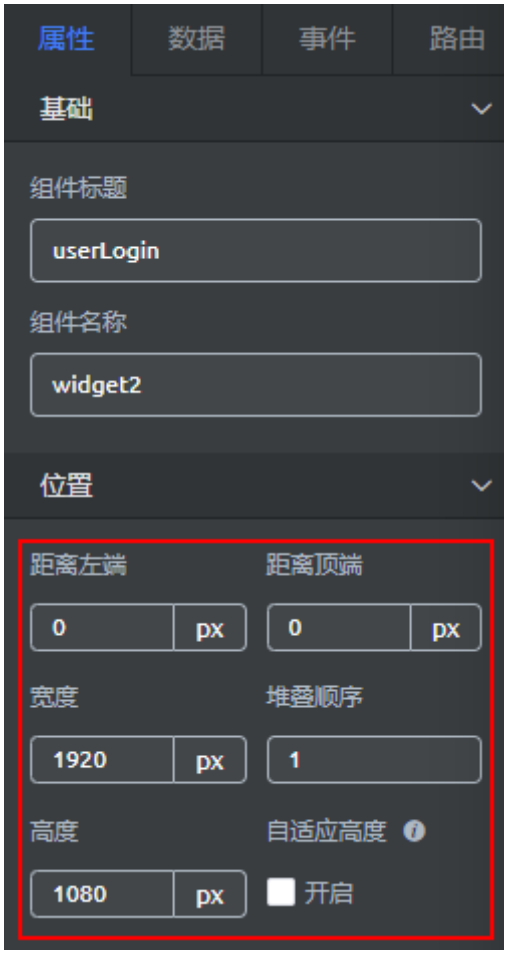
图 11-34 拖拽组件到编辑区



步骤2 设置自定义组件“userLogin”的位置属性。

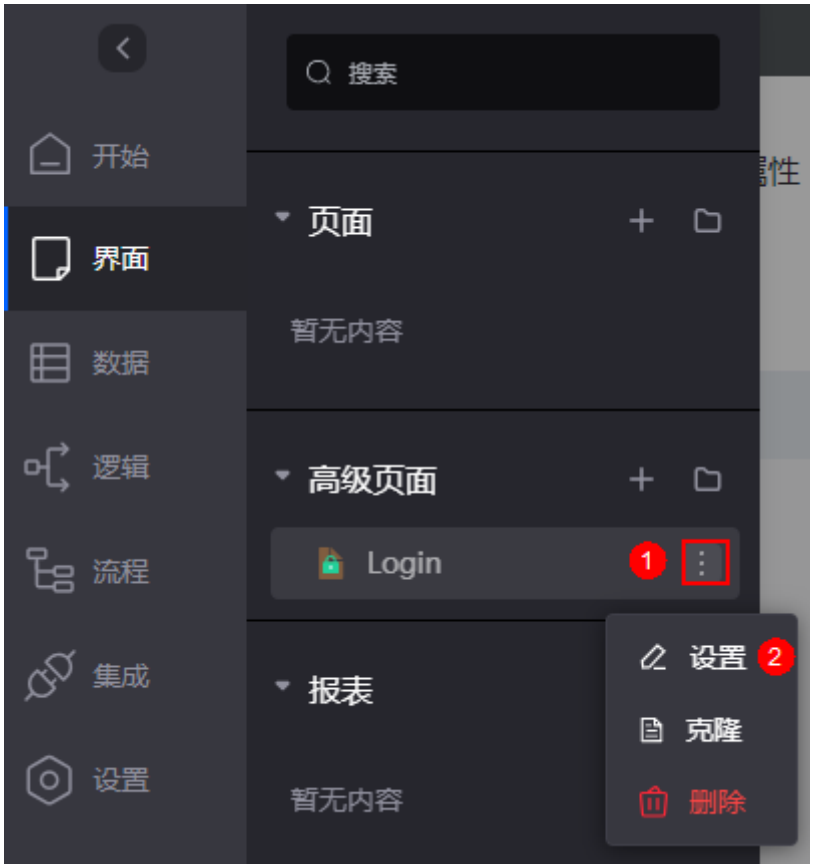
1. 将“userLogin”组件拖拽到页面编辑区后，会在右侧显示该组件的属性配置面板。
2. 在“位置”中，设置“距离左端”、“距离顶端”为“0”，“宽度”为“1920”，“高度”为“1080”。

图 11-35 设置组件位置



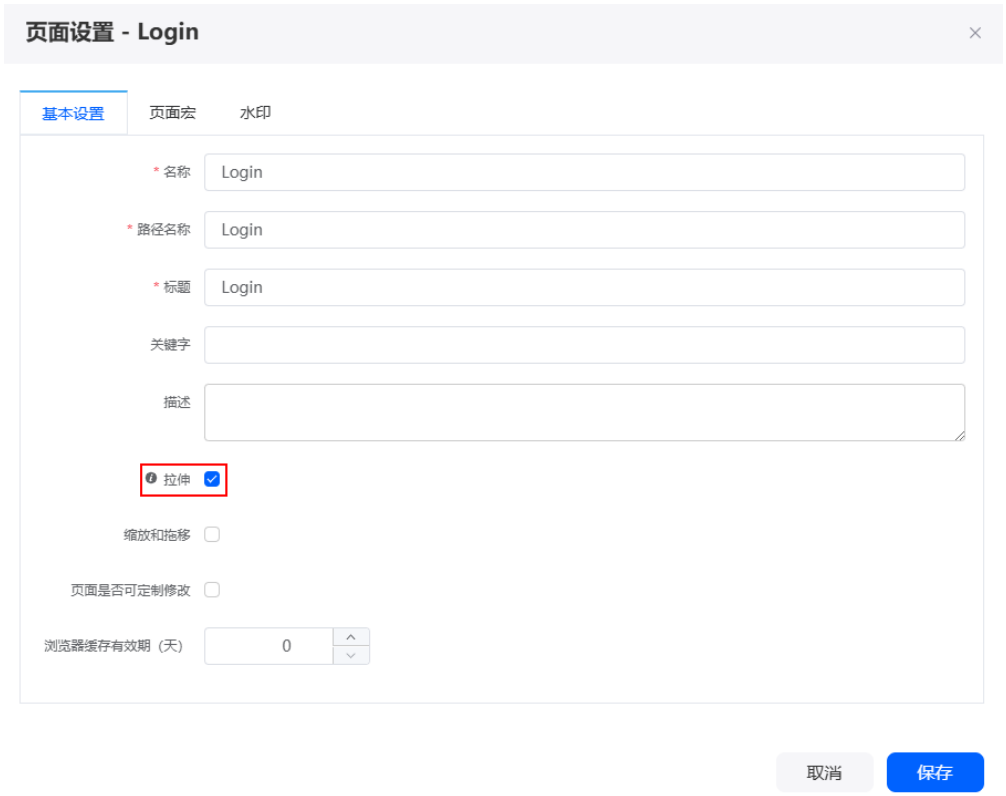
3. 组件位置、样式等属性修改完成后，单击页面上方的，保存页面修改。
4. 在“界面 > 高级页面”中，将鼠标放在“Login”上，单击，选择“设置”。

图 11-36 选择设置



5. 在页面设置中，选中拉伸属性，单击“保存”。

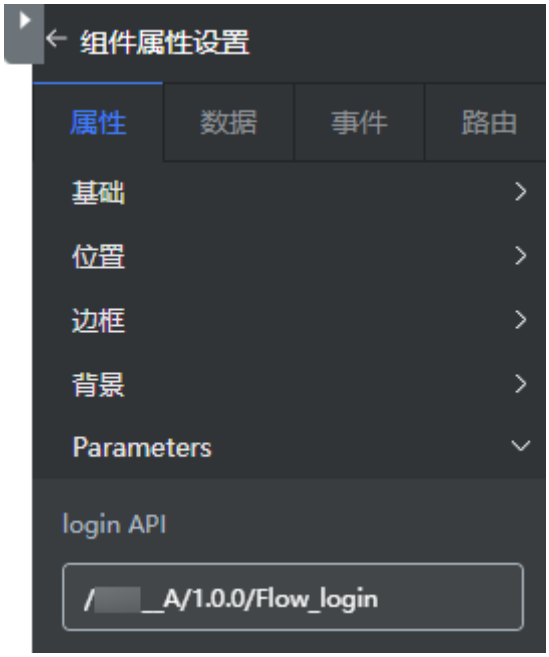
图 11-37 设置页面拉伸



步骤3 设置自定义组件的桥接器。

1. 选中“userLogin”组件，在“属性 > Parameters”中，设置“login API”为“/*命名空间*__A/1.0.0/Flow_login”。

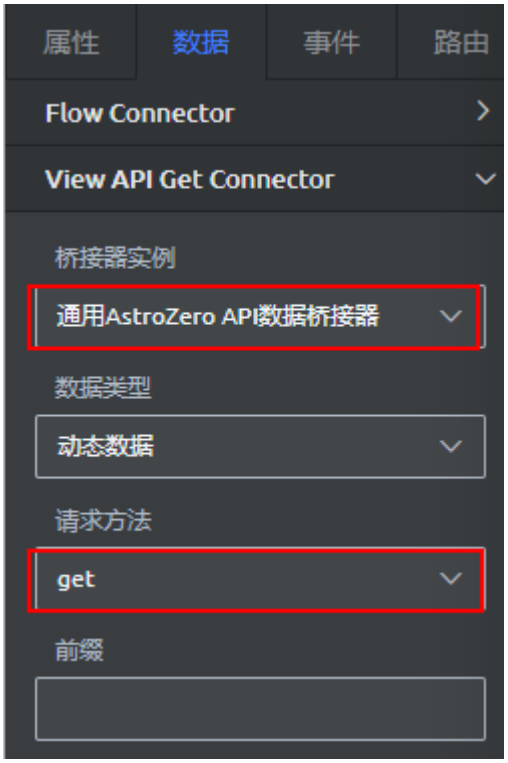
图 11-38 添加登录服务 URL



“login API”为登录接口的URL后半段，“命名空间”请根据实际情况配置，“A”为应用名，登录接口是在**业务用户登录页后端逻辑开发实施步骤**中创建的。

2. 在“数据”页签，单击“View API Get Connector”，设置“桥接器实例”为“通用AstroZero API数据桥接器”，“数据类型”为“动态数据”，“请求方法”为“get”，如下图所示。

图 11-39 设置桥接器



3. 参考上一步，分别设置下图中框选的桥接器实例。

图 11-40 设置其他桥接器

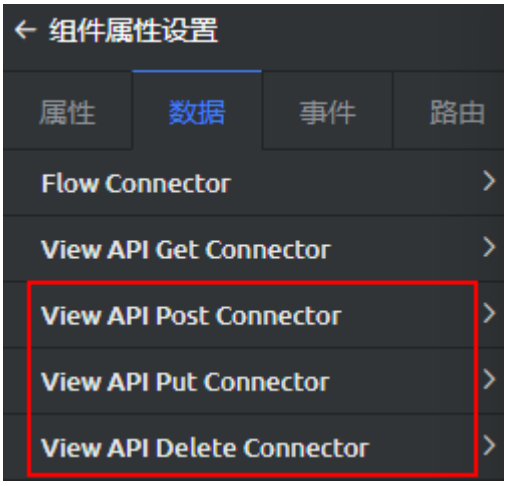


表 11-8 桥接器实例配置

数据名	桥接器实例	数据类型	Request Method
View API Post Connector	通用AstroZero API数据桥接器	动态数据	Post
View API Put Connector	通用AstroZero API数据桥接器	动态数据	Put
View API Delete Connector	通用AstroZero API数据桥接器	动态数据	Delete

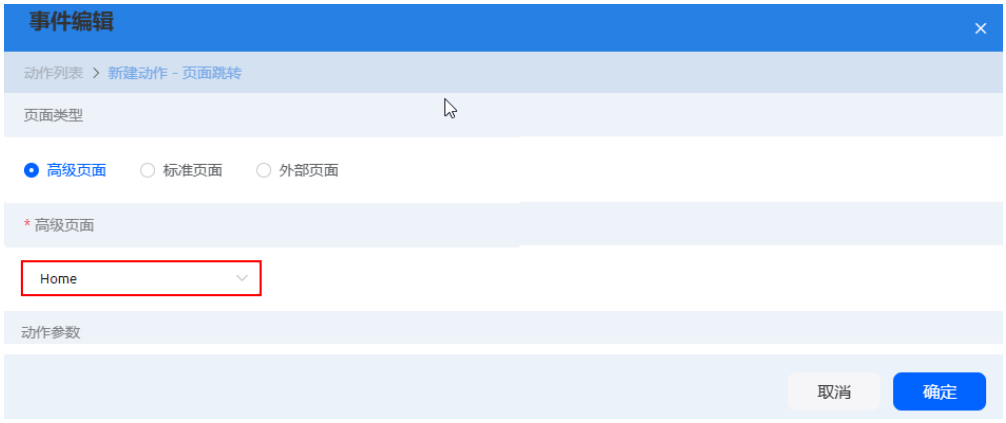
步骤4 设置自定义组件“userLogin”的“事件”，使自定义组件与其他页面关联。

图 11-41 需要配置的事件

1. 单击“go Homepage”后的齿轮图标，进入事件编辑页面。
2. 单击“新建动作”，选择“默认 > 页面跳转”，进入跳转编辑页面。
3. 选择需要跳转的页面，单击“确定”。

   这里选择的高级页面是之前创建并发布的“Home”，在实际开发过程中，可以根据业务需要，选择跳转到需要的页面，或者执行自定义的业务逻辑。

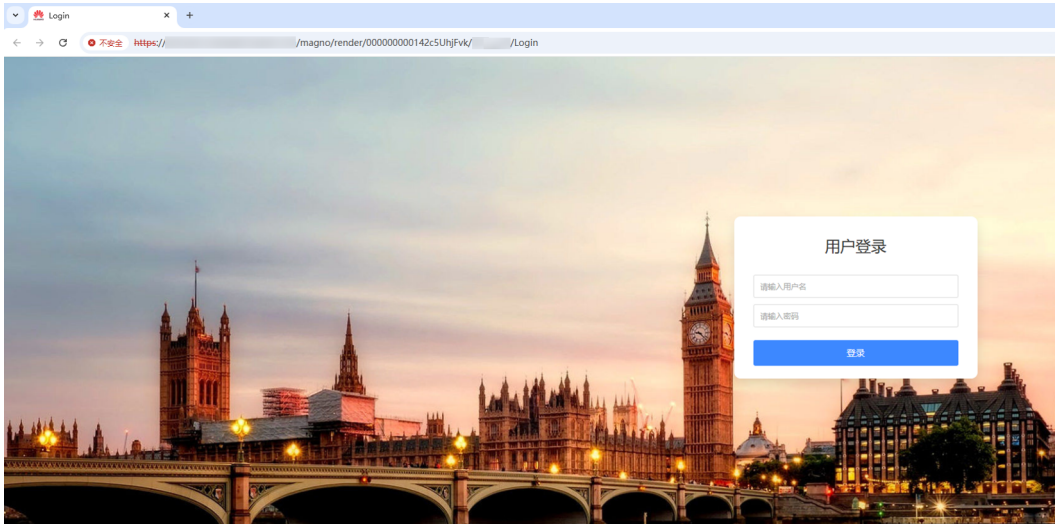
图 11-42 编辑页面跳转



步骤5 在上述步骤完成后，单击Login页面上方的，保存页面，再单击，发布页面。

步骤6 页面发布成功后，单击，即可预览发布的登录页面。

图 11-43 预览的登录页面



说明

当前登录页中，输入业务用户账号及密码，单击“登录”的登录逻辑是通过“自定义登录”组件，调用用户登录服务编排完成的。

步骤7 在预览的登录页面中，输入配置了权限的业务用户的账号密码，单击登录后，如果页面跳转到“Home”页面，则业务用户登录成功。

----结束