

Astro 企业应用

用户指南

文档版本 01
发布日期 2024-07-03



版权所有 © 华为云计算技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI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。

目录

1 新手指引	1
1.1 入门必读	1
1.2 申请 AstroPro 公测	2
1.3 购买基础版实例	5
1.4 购买专业版实例	6
1.5 登录 AstroPro 界面	9
2 前端应用管理	12
2.1 初识应用设计器	12
2.2 设计前端应用流程	17
2.3 应用管理	23
2.4 页面管理	24
2.5 使用组件	31
2.6 属性设置	37
2.7 样式设置	38
2.7.1 通过样式面板配置样式	38
2.7.2 直接编写样式代码	44
2.7.3 类名管理	46
2.7.3.1 新增类名	46
2.7.3.2 修改类名	46
2.7.3.3 删除类名	47
2.7.3.4 选择已有类名	48
2.7.4 使用状态选择器	48
2.7.5 行内绑定状态变量	49
2.8 高级设置	52
2.8.1 条件渲染	52
2.8.2 循环渲染	54
2.8.3 绑定事件	57
2.9 查看大纲树	58
2.10 数据源管理	59
2.10.1 获取数据源远程字段	59
2.10.2 添加静态数据	60
2.10.3 使用数据源	61
2.11 使用工具类方法	64

2.11.1 添加工具类.....	64
2.11.2 npm utils 使用示例.....	67
2.11.3 function utils 使用示例.....	69
2.12 国际化资源管理.....	70
2.13 使用 JS 面板.....	75
2.14 变量管理.....	78
2.14.1 添加页面变量.....	78
2.14.2 添加全局变量.....	79
2.15 生成业务代码.....	80
3 后台应用管理.....	82
3.1 创建企业核心应用.....	82
3.1.1 了解构建流程.....	82
3.1.2 步骤 1：创建项目.....	83
3.1.3（可选）步骤 2：创建服务组.....	85
3.1.4 步骤 3：添加服务.....	86
3.1.5 步骤 4：编辑服务.....	89
3.1.6 步骤 5：生成服务代码.....	91
3.2 项目管理.....	94
3.2.1 新建项目.....	94
3.2.2 编辑项目.....	97
3.2.3 删除项目.....	99
3.2.4 角色管理.....	99
3.2.4.1 了解 AstroPro 中角色.....	99
3.2.4.2 为用户添加工作空间级角色.....	102
3.2.4.3 为用户添加项目级角色.....	106
3.3 服务组管理.....	111
3.3.1 新建服务组.....	112
3.3.2 编辑服务组.....	113
3.3.3 删除服务组.....	114
3.4 服务管理.....	114
3.4.1 了解服务创建流程.....	114
3.4.2 新增一个服务.....	115
3.4.3 编辑服务.....	117
3.4.3.1 步骤 1：基本配置.....	118
3.4.3.2 步骤 2：框架配置.....	119
3.4.3.3 步骤 3：生成策略.....	123
3.4.3.4 步骤 4：业务设计.....	127
3.4.3.5 步骤 5：服务依赖.....	130
3.4.4 生成服务代码.....	131
3.4.5 查看服务详情.....	132
3.4.6 升级 API 版本.....	132
3.4.7 重新编译服务.....	134

3.4.8 复制服务.....	135
3.4.9 删除服务.....	136
3.4.10 导出元数据.....	138
3.5 服务依赖管理.....	139
3.5.1 新增依赖服务.....	139
3.5.2 查看服务依赖.....	142
3.5.3 删除服务依赖.....	142
3.6 对象详解.....	142
3.6.1 BO.....	143
3.6.2 Abstract BO.....	144
3.6.3 Value Object.....	146
3.6.4 对象间关系.....	149
3.6.4.1 一对多.....	149
3.6.4.2 多对多.....	150
3.6.4.3 聚合.....	152
3.6.4.4 树递归.....	153
3.6.4.5 继承.....	155
4 AstroPro 学堂.....	157
4.1 如何自定义 DTO.....	157
4.2 如何定义脱敏规则.....	159
4.3 如何为对象自定义 API.....	160
4.4 如何为对象添加固定字段.....	161
4.5 如何为对象添加枚举字段.....	163
4.6 服务开发框架详解.....	164
4.6.1 整体结构介绍.....	165
4.6.2 单 Module.....	166
4.6.3 base/service.....	168
4.6.4 DDD.....	171
4.7 AstroPro-SDK 版本变更与下载.....	173

1 新手指引

1.1 入门必读

Astro企业应用（AstroPro，简称AstroPro）通过元数据多租、高低代码协同能力，快速构建企业核心业务应用，提高多层级/多分支企业的应用构建效率。不过，要熟练使用AstroPro，还需要进行一些基础和深入的学习。这里，我们为您总结了用户咨询的高频问题和搭建经验，并制定了一套完整的学习计划，希望对您的使用有所引导及帮助。

1、快速了解 AstroPro

通过AstroPro产品介绍，快速了解什么是AstroPro。单击[产品介绍](#)，了解更多。

2、创建一个订单系统，快速体验 AstroPro

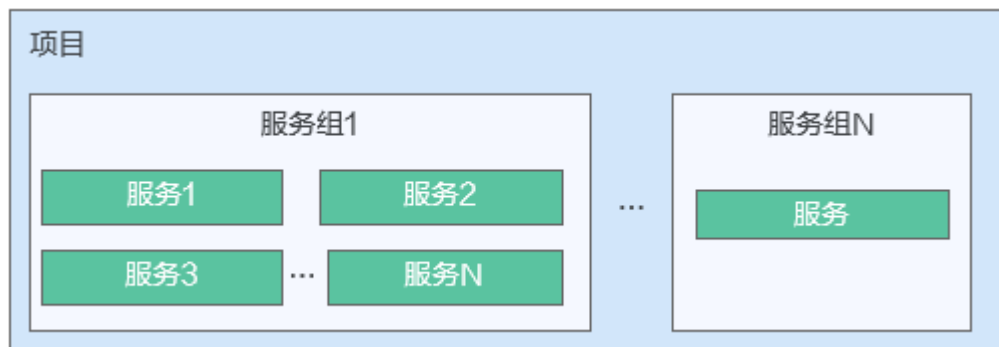
通过[小试牛刀](#)，帮助您快速熟悉使用AstroPro生成服务代码的过程。

3、了解 AstroPro 中的项目、服务组与服务之间的关系

项目是使用AstroPro核心业务的入口。服务组用于对项目中的服务进行分组，一般一个分组对应一个研发团队。服务组创建后，即可为项目添加服务。服务是业务概念，即提供某种服务的某个进程。每一个服务都具有自主运行的业务功能，对外开放不受语言限制的API，多个服务组成应用程序。

在AstroPro中，项目、服务组和服务之间的关系，如[图1-1](#)所示。

图 1-1 项目、服务组与服务的关系



4、熟悉如何进行业务设计

在AstroPro中，用户通过业务设计，可生成高可用、高可靠、以及安全稳定的企业级IT应用框架。

- **对象**：对象可以理解为数据库中创建的一个表。每个对象对应一张数据库表，用于保存业务系统需要的配置数据和业务数据。对象可以存储组织或业务特有的数据，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，对象就是电影中的各个角色，需要勾勒角色的外貌、性格特点、人物关系和所经历的剧情。

AstroPro提供了BO、Abstract BO和Value Object三种类型的对象，请根据业务需求进行选择。

- **BO**：业务对象，业务对象映射到服务中的一个实体，对应数据库中的一张表。
- **Abstract BO**：抽象对象，不能实例化，没有对应的数据库表，需要和业务对象有个继承的操作。例如，业务对象A继承一个抽象对象B，则B中的字段都会被A继承过来。
- **Value Object**：值对象，不能单独存在，需要和业务对象建立聚合的关系。
- **对象间关系**：关系描述了不同元素之间的关联和联系，在AstroPro中您可以定义一对多、多对多、聚合和继承等关系。

1.2 申请 AstroPro 公测

使用说明

在使用AstroPro前，请先申请公测。公测申请通过后，才可正常使用AstroPro。

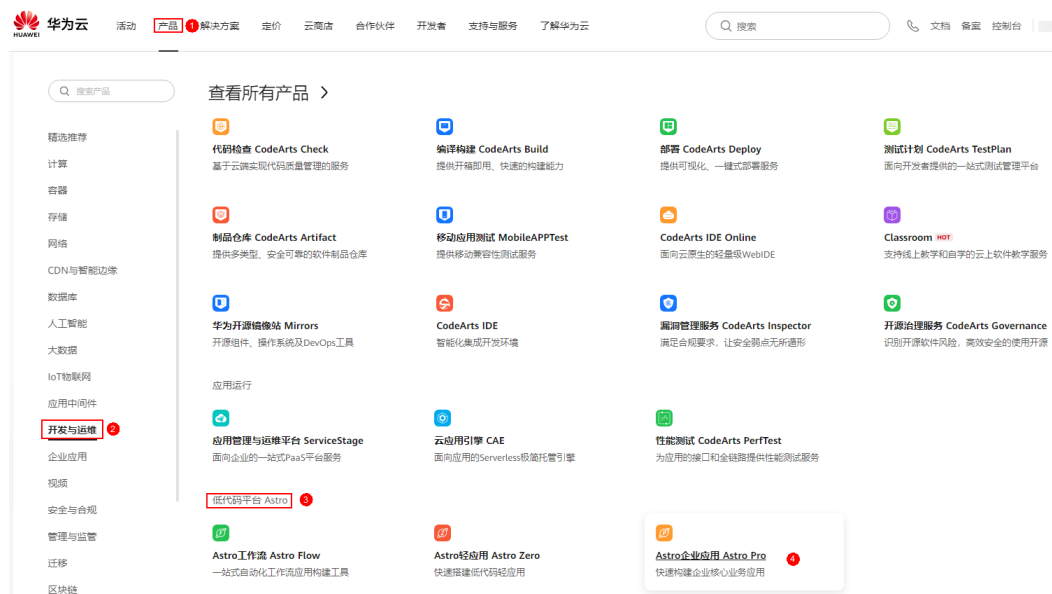
操作步骤

步骤1 登录[华为云官网](#)。

如果您没有华为账号，请立即注册，相关操作请参见[如何进行账号注册](#)。注册后参考[个人账户如何完成实名认证](#)或[企业账号如何完成实名认证](#)完成个人或企业账号实名认证。

步骤2 在“产品”中，选择“开发与运维 > 低代码平台 Astro > Astro企业应用 Astro Pro”。

图 1-2 选择 Astro 企业应用



步骤3 在AstroPro产品介绍页，单击“申请公测”。

步骤4 设置企业规模、联系人等信息，勾选公测试用服务协议，单击“申请公测”。

图 1-3 填写申请公测信息

申请公测

申请的公测产品/服务

Astro Pro 公测

申请信息

* 企业规模: 50人以下 | 50-100人 | 100-200人 | 200-500人 | 500人以上

* 研发人员比例: 0%~10%

* 应用场景: web

* 业务当前阶段: 开发 | 测试 | 试运行 | 正式运营

* 业务描述: AstroPro资料写作

* 联系人: [输入框]

* 联系电话: +86(中国大陆) | 186 [输入框]

电子邮箱: [输入框].com

同意 《公测试用服务协议》

申请公测

步骤5 申请提交后，单击“前往我的公测”，查看审批结果。

图 1-4 单击前往

申请公测

提交成功

我们已收到您的申请，5个工作日内审核结果将发送到您的邮箱和手机，感谢您的支持。

前往我的公测

步骤6 公测申请审批通过后，单击“前往控制台”，即可进入Astro企业应用服务控制台。

图 1-5 前往控制台

产品服务	公测状态	申请时间	审批状态	审批说明	操作
Astro Pro 公测	公测中	2023/12/14 08:47:15 GMT+08:00	审批通过	-	修改公测联系人信息 前往控制台

----结束

1.3 购买基础版实例

使用说明

在使用AstroPro前，您需要购买一个AstroPro实例。AstroPro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。

基础版实例当前为免费使用，仅能满足个人及创业团队的基础需求。基础版和专业版实例支持的特性差异，请参见[产品规格差异](#)。为了便于您更好的了解AstroPro，本指南中操作及截图均以专业版实例为例进行介绍。

前提条件

公测申请已审批通过。如何申请公测，请参见[申请AstroPro公测](#)。

操作步骤

步骤1 进入[购买Astro企业应用实例页面](#)。

步骤2 “实例规格”选择“基础版”，设置购买时长，单击“立即购买”。

图 1-6 选择购买规格



步骤3 确认订单无误后，单击“去支付”。

图 1-7 确认订单

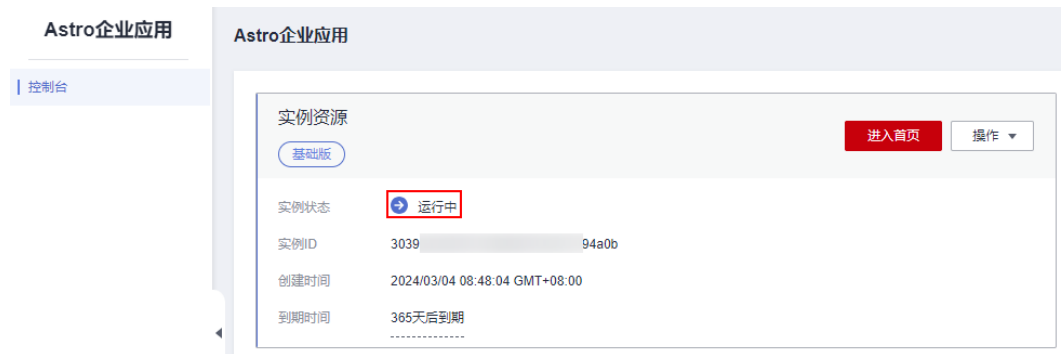


步骤4 选择支付方式，单击“确认付款”，完成支付。

步骤5 订单支付成功后，单击“返回Astro企业应用控制台”。

在Astro企业应用控制台中，可以查看到Astro企业应用的实例状态。当“实例状态”变为“运行中”时，说明实例已安装好，可以正常使用Astro企业应用。

图 1-8 查看实例状态



----结束

1.4 购买专业版实例

使用说明

在使用AstroPro前，您需要购买一个AstroPro实例。AstroPro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。

相对于基础版实例，专业版实例提供了更多的应用及实体功能，可满足中大型企业的复杂管理需求。除此之外，购买专业版实例时或购买实例后，支持对资源进行扩容。专业版和基础版实例支持的特性差异，请参见[产品规格差异](#)。为了便于您更好的了解AstroPro，本指南中操作及截图均以专业版实例为例进行介绍。

前提条件

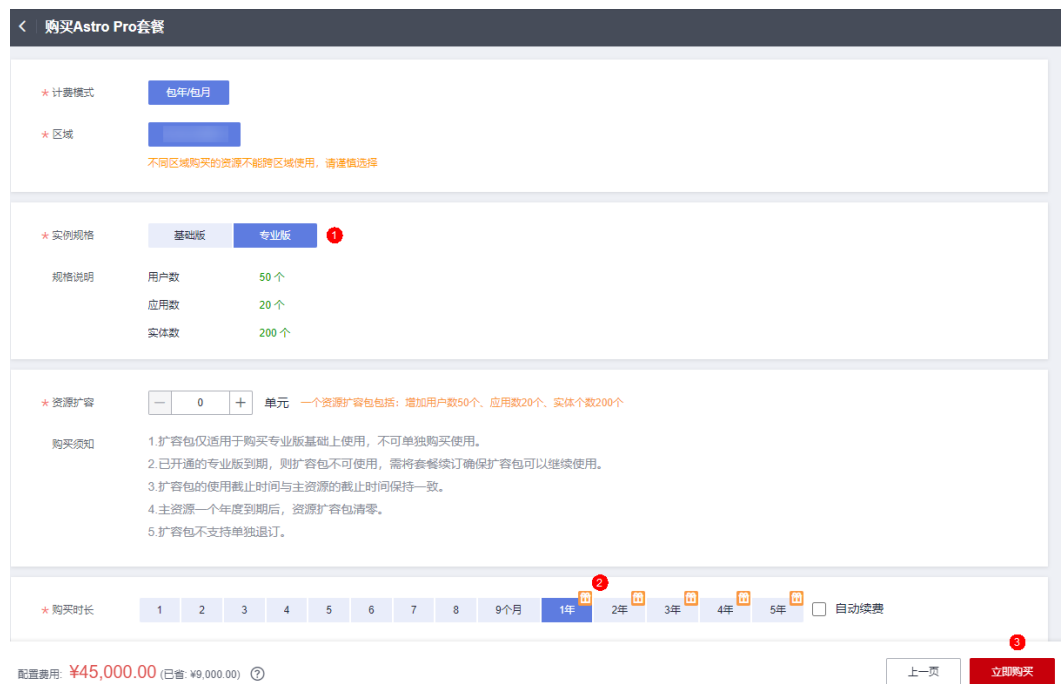
- 已参考[申请AstroPro公测](#)中操作，申请公测，且公测申请已审批通过。
- 确保账户有充足的费用。如何为账户充值，请参见[账户充值](#)。

购买实例

步骤1 进入[购买Astro企业应用实例页面](#)。

步骤2 “实例规格”选择“专业版”，其他参数按需进行设置，单击“立即购买”。

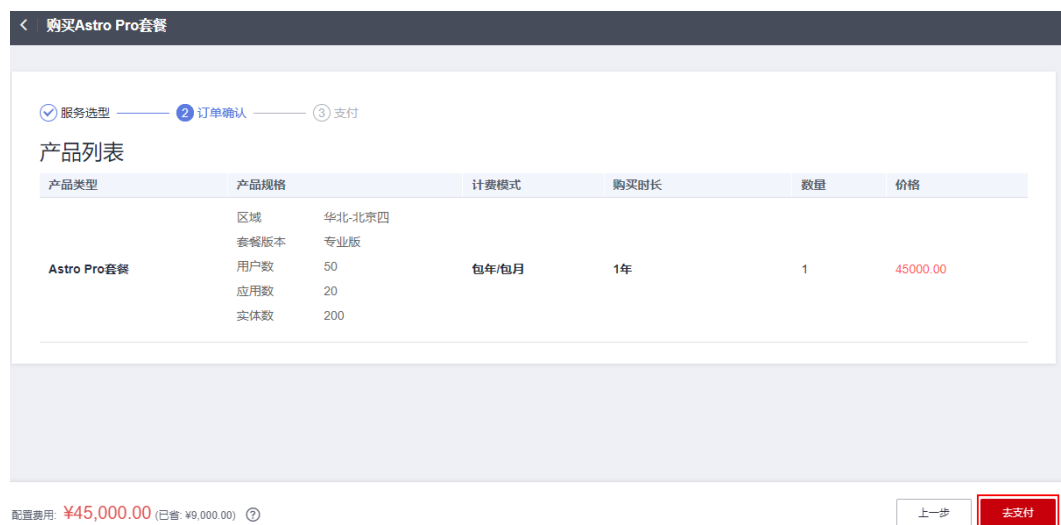
图 1-9 选择购买规格



步骤3 确认订单无误后，单击“去支付”。

以购买专业版一个月为例，若为基础版，此处配置费用为“0”。

图 1-10 确认订单

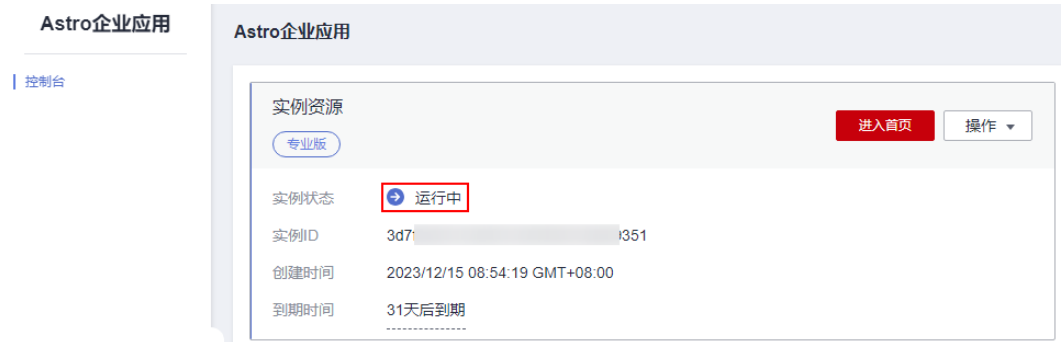


步骤4 选择支付方式，单击“确认付款”，完成支付。

步骤5 订单支付成功后，单击“返回Astro企业应用控制台”。

在Astro企业应用控制台中，可以查看到Astro企业应用的实例状态。当“实例状态”变为“运行中”时，说明实例已安装完成，可以正常使用Astro企业应用。

图 1-11 查看实例状态



----结束

购买扩容包

扩容包仅适用于在购买专业版的基础上使用，不可单独购买和退订。扩容包的使用截止时间与主资源的截止时间保持一致。

步骤1 参考[购买实例](#)中操作，购买AstroPro专业版实例。

步骤2 在已购买的实例中，单击“操作”，选择“扩容”。

图 1-12 选择扩容



步骤3 在购买扩容包页面，设置购买数量，单击“立即购买”。

一个资源扩容包，包含用户数50个、应用数20个、实体个数200个，请按需设置购买数量。

图 1-13 购买扩容包



步骤4 选择支付方式，单击“确认付款”，完成订单支付。

步骤5 返回AstroPro服务控制台，在已购买的实例中，单击“进入首页”，查看扩容后的资源规格。

图 1-14 查看扩容后规格



---结束

1.5 登录 AstroPro 界面

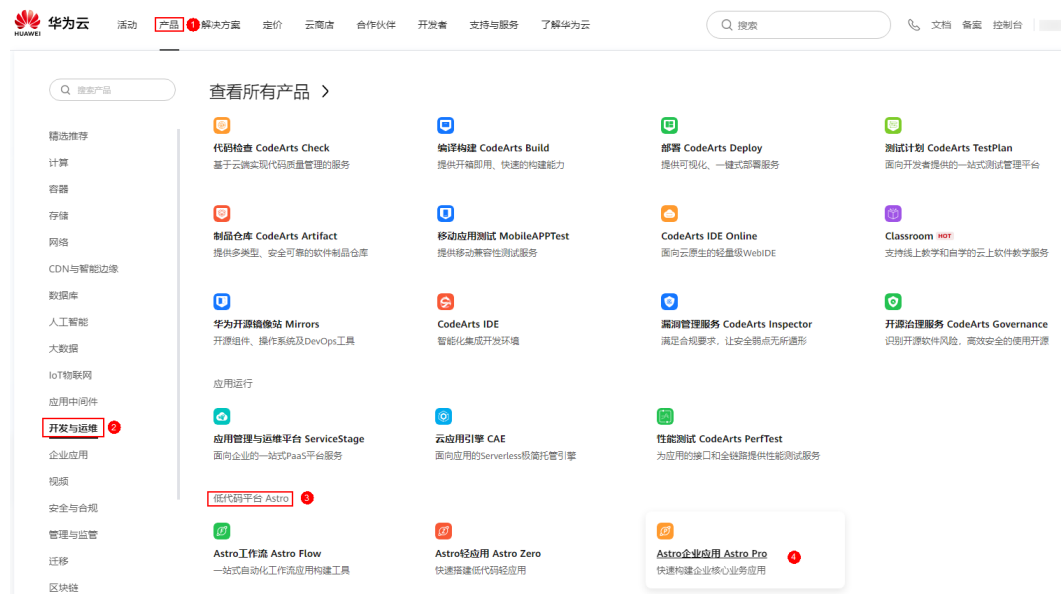
前提条件

- 已参考[申请AstroPro公测](#)中操作，申请公测。
- 已参考[购买基础版实例](#)或[购买专业版实例](#)中操作，购买AstroPro实例。

操作步骤

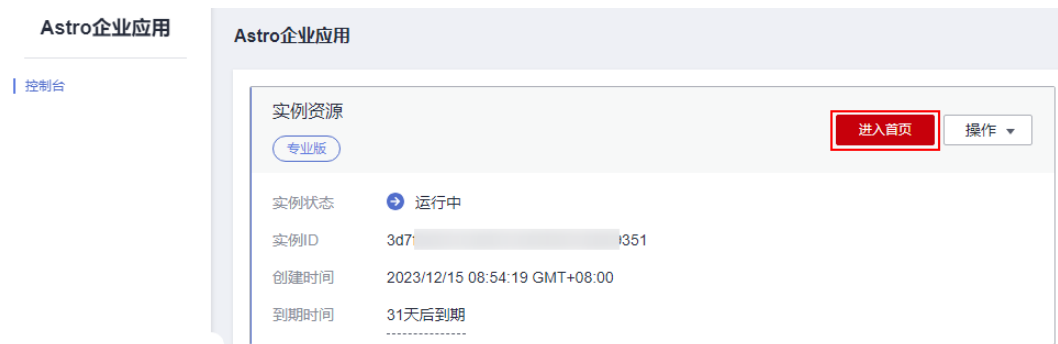
- 步骤1** 使用华为账号，登录[华为云网站](#)。
- 步骤2** 在顶部导航栏右侧单击“控制台”，进入华为云控制台。
- 步骤3** 在“产品”中，选择“开发与运维 > 低代码平台 Astro > Astro企业应用 Astro Pro”。

图 1-15 选择 Astro 企业应用



- 步骤4** 在AstroPro服务控制台的首页中，单击已购买实例中的“进入首页”，即可进入AstroPro界面。

图 1-16 AstroPro 服务控制台



首次进入首页时，请勾选AstroPro隐私协议及服务声明，如[图1-17](#)所示。

图 1-17 勾选隐私协议及服务声明



图 1-18 AstroPro 界面



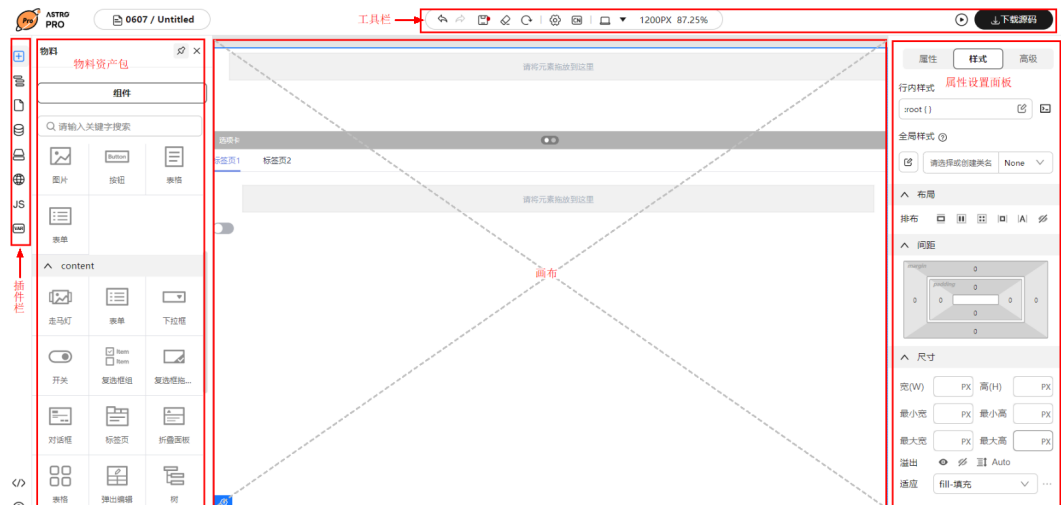
----结束

2 前端应用管理

2.1 初识应用设计器





AstroPro设计器可以分为**顶部工具栏**、左侧插件栏、中间画布区、右侧属性设置面板几个主要界面模块。






图 2-1 AstroPRO 设计器











顶部工具栏

设计器顶部的工具栏从左到右包括：

- ：撤销上一步的操作。
- ：恢复上次撤销的内容。
- ：保存当前页面数据。
- ：清空当前画布内容。

- ：刷新当前页面数据，如果页面未保存，页面配置将恢复到上一次保存的状态。
- ：单击后可展开左侧插件栏的设置插件，设置当前页面属性。
- ：单击可进行画布中英文切换。
- ：多终端工具切换，可切换画布宽度。
- ：页面预览，打开新Tab预览当前页面。
- 下载源码：将当前页面数据转换并下载代码到本地。

左侧插件栏

- ：物料插件，提供设计所需组件，拖拽组件至画布中进行页面设计，具体操作请参考[使用组件](#)。
- ：大纲树插件，可[查看页面大纲树](#)。
- ：页面管理工具插件，可以[新增文件夹](#)，可以[新增文件](#)，以及对页面或者文件的增删改操作。
- ：数据源管理插件，可用于来配合画布上的组件渲染，具体操作请参考[使用数据源](#)。
- ：资源管理插件，将一些可复用的公共函数编写到工具类中，也可以将一些npm包引用到工具类中，供后续调用，具体操作请参考[使用工具类方法](#)。
- ：国际化插件，可[添加国际化词条](#)，实现中英文切换。
- ：JS方法插件，您可以通过[使用JS面板](#)编写自己的代码，从而实现较为复杂的业务场景。
- ：状态管理插件，可[添加页面变量](#)和[添加全局变量](#)，供页面及应用使用。


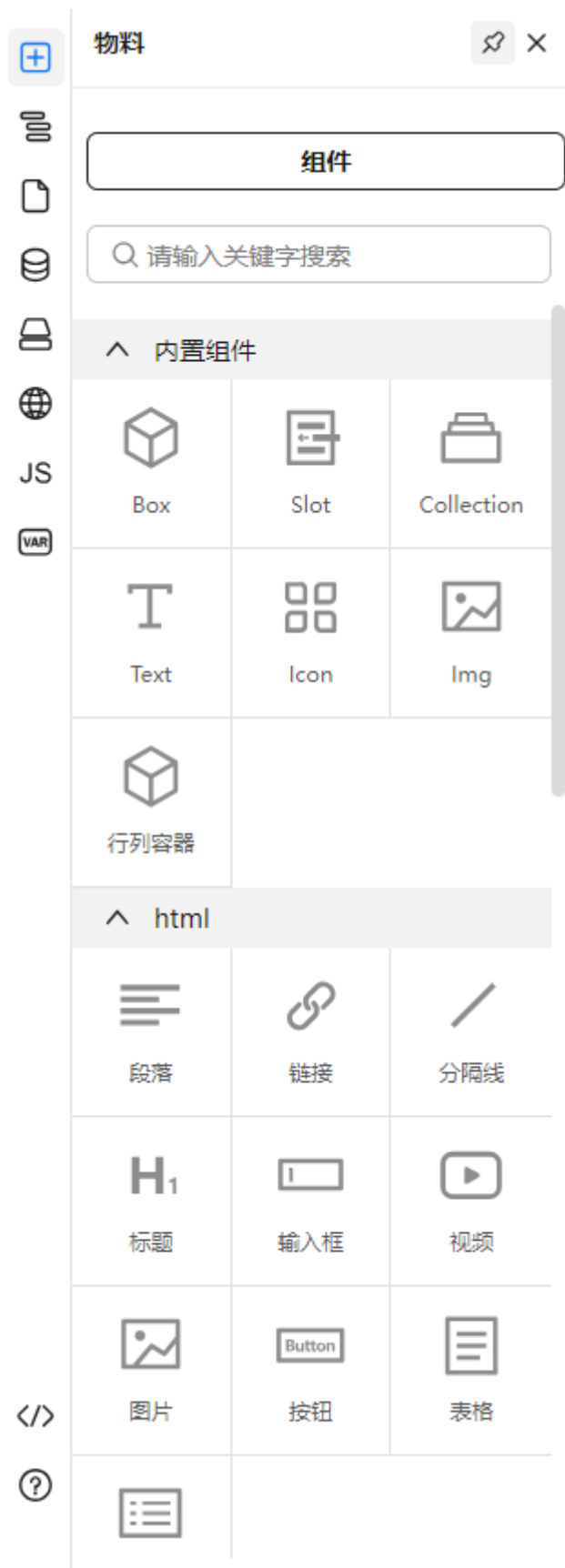
插件单击之后会向右展开对应插件的设置面板。例如，单击 ，将展开物料资产包。

图 2-2 展开物料资产包



中间画布区

中心画布位于设计器中央，是可视化设计的核心模块，可以在插件栏的物料面板中，将组件拖拽至中心画布内，然后单击画布选中组件，修改组件的属性、样式、绑定事件等。

右侧属性设置面板

右侧设置面板分为属性设置、样式设置和高级设置面板。

- 属性设置，设置组件的属性，比如按钮组件的大小、文案、按钮类型等组件提供配置的属性。

图 2-3 基本属性设置

属性 样式 高级

^ 基础信息

id

className

ref

src

自定义属性 +

无数据

- 样式设置，设置组件的样式，提供直接配置样式，也可以编写CSS代码配置样式。

图 2-4 样式设置



- 高级设置，设置组件是否渲染、绑定单击事件、设置组件是否循环渲染等。

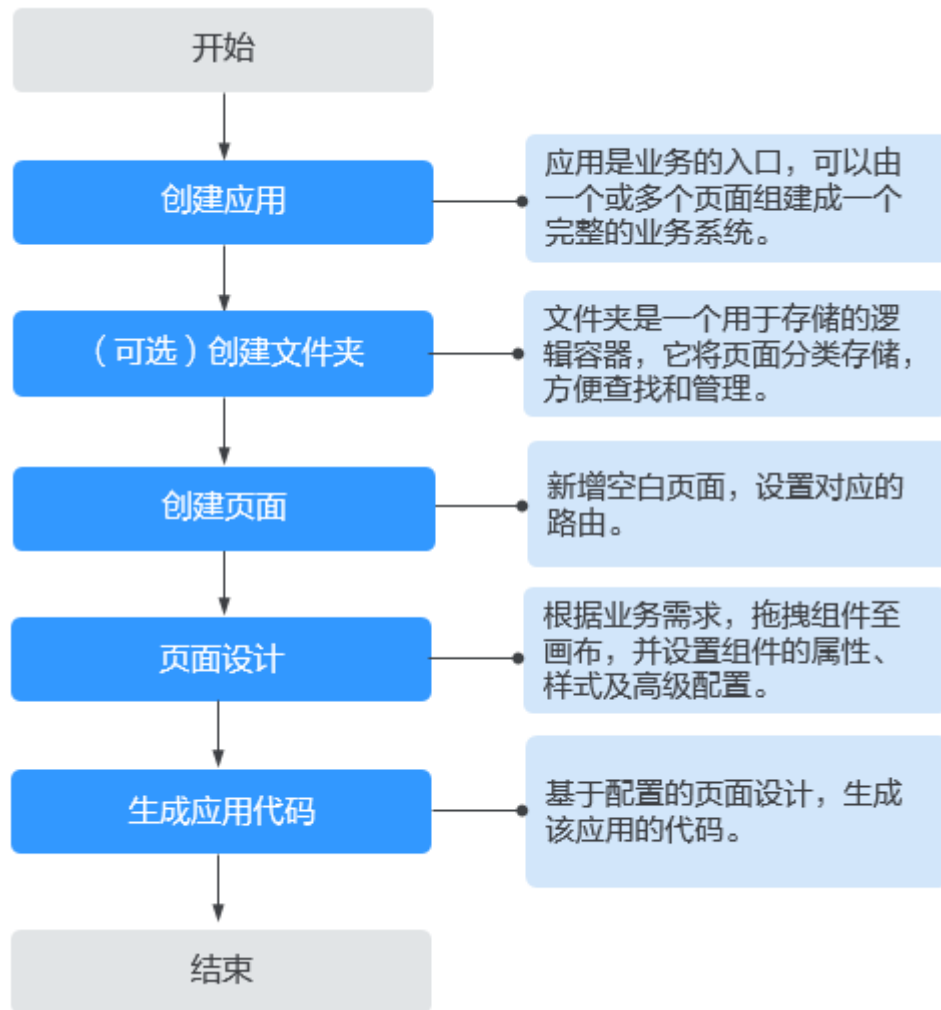
图 2-5 高级设置



2.2 设计前端应用流程

应用可以由一个或多个页面组成一个完整的业务系统。在AstroPro中，通过创建应用、（可选）创建文件夹、创建页面、页面设计和生成应用代码五步，即可快速完成应用设计，具体流程如[图2-6](#)所示。

图 2-6 创建应用



步骤一：新建一个应用

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”，单击“创建应用”。

步骤3 输入应用名称及应用描述。

图 2-7 创建应用




步骤4 单击“确定”，完成应用创建。

----结束

步骤二：新建文件夹

步骤1 单击应用模块内的“应用开发”，进入设计器。

步骤2 在左侧插件栏中，单击 ，进入“页面管理”页面。

步骤3 单击“页面管理”的新增文件夹按钮。

步骤4 设置基本属性，如输入文件夹ID及设置路由。

图 2-8 创建文件夹



步骤5 单击“保存”，完成文件夹创建。

----结束

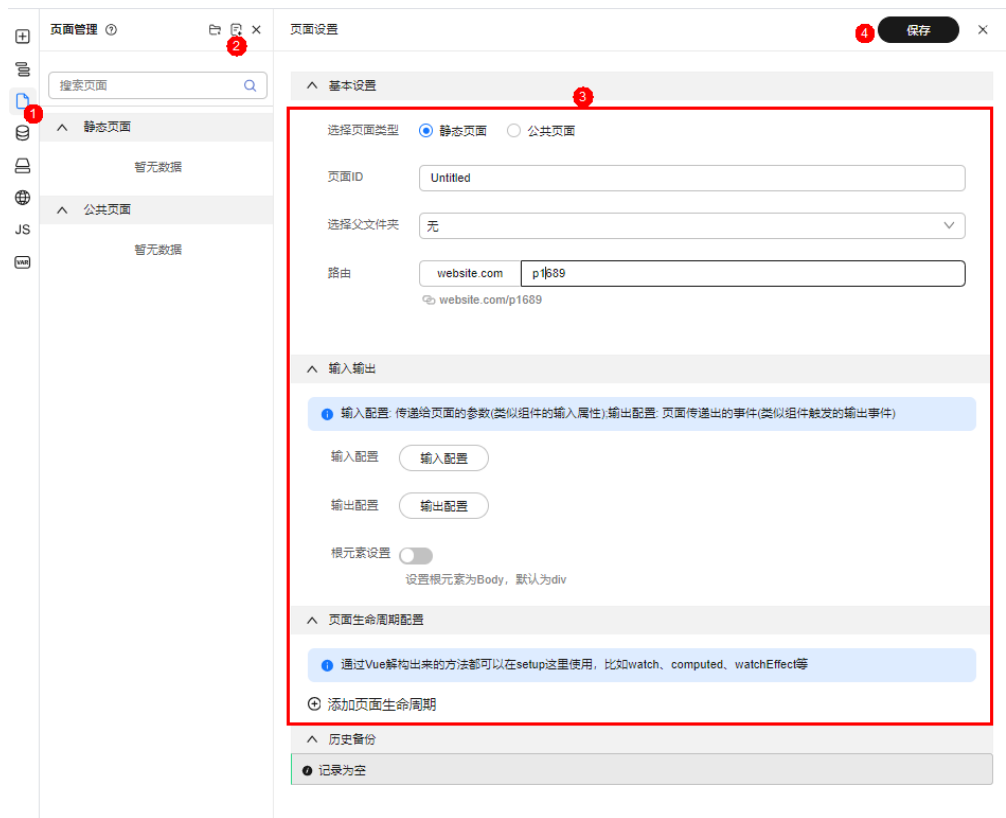
步骤二：新建一个页面

步骤1 单击“页面管理”的新增页面按钮。

步骤2 设置页面基本属性。

- 选择页面类型：可选“静态页面”或“公共页面”。
- 页面名称：只允许包含英文字母，且以大写开头驼峰格式，如DemoPage。
- 选择文件夹：下拉框中选择文件夹名称。
- 路由：输入路由信息，只允许包含英文字母、数字、下划线、中划线和正斜杠组成，且以英文字母开头。

图 2-9 创建应用



步骤3 单击“保存”。

步骤4 在弹框中输入历史备份信息，单击“确定”。

----结束

步骤二：页面设计

前端应用由一个或多个组件拼装而成。在左侧插件栏打开物料资产包，选择合适的组件，拖拽到中间画布中。在画布选中组件，设置组件的属性、样式、以及绑定事件。

图 2-10 添加组件

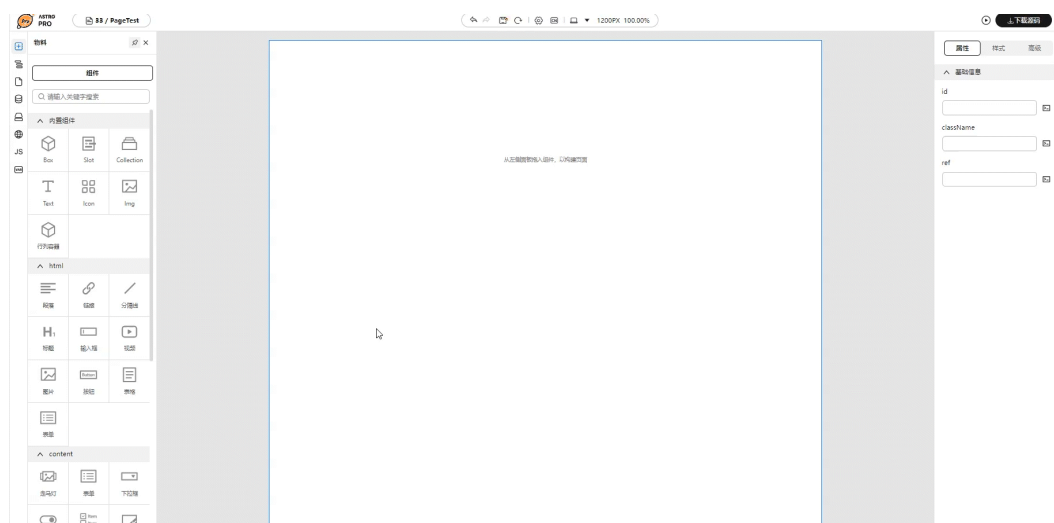


图 2-11 添加组件

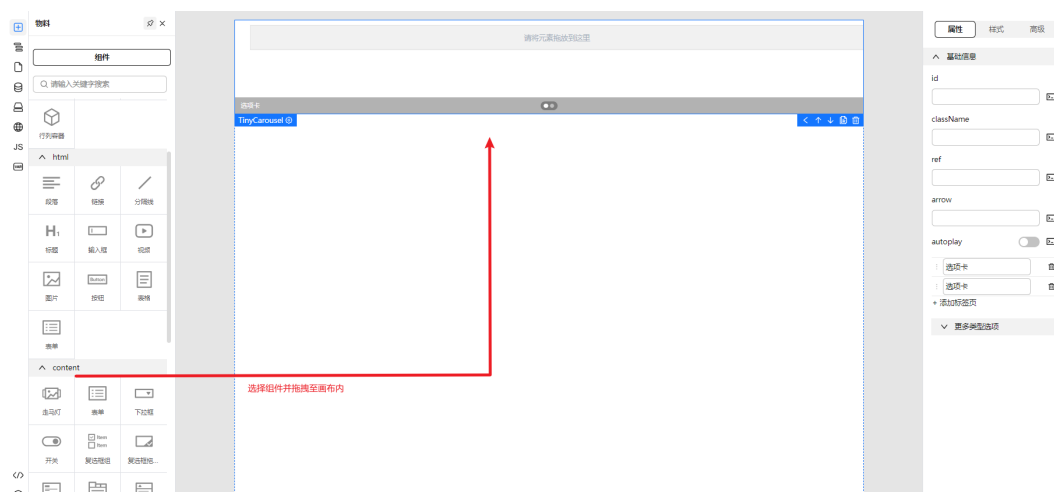


图 2-12 设置组件基本属性

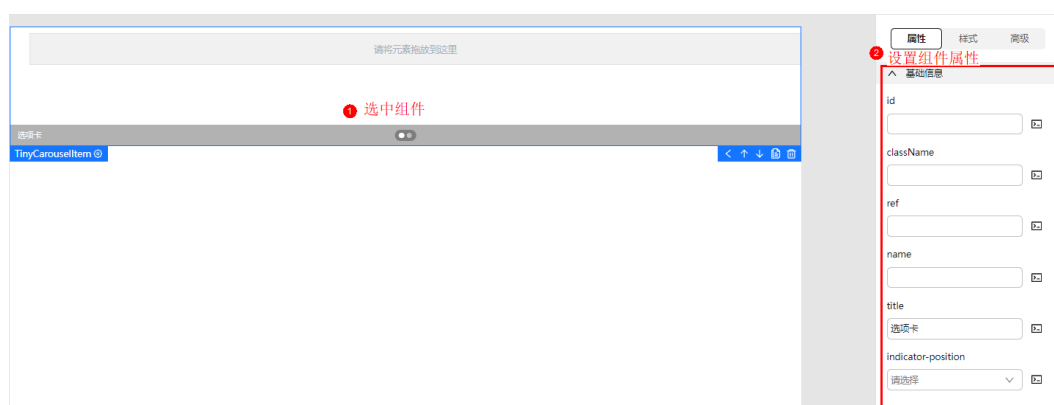


图 2-13 设置组件样式

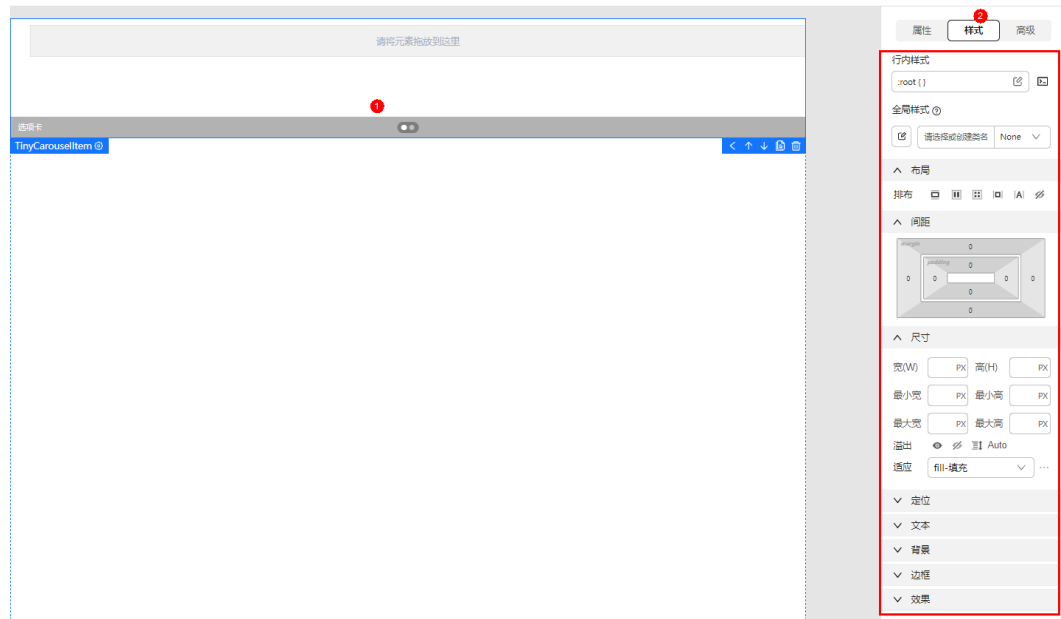
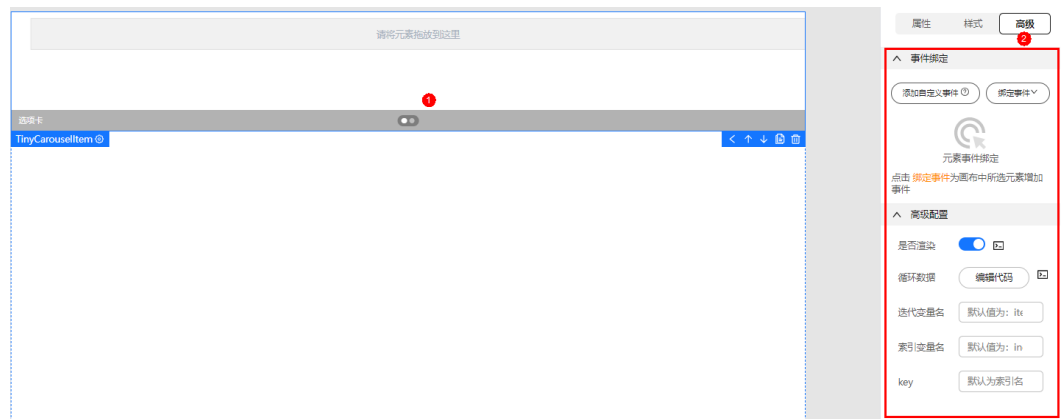


图 2-14 设置组件高级属性



步骤 3: 生成代码

根据配置的页面设计，生成应用的基本代码。代码生成后，会下载至本地，供您使用。

- 步骤1** 单击顶部工具栏的“下载源码”按钮。
- 步骤2** 选择下载路径。
- 步骤3** 选择生成到本地的文件。

图 2-15 选择生成文件



步骤4 单击“确定”，代码将下载至本地路径。

----结束

2.3 应用管理

创建应用

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”，单击“创建应用”。

步骤3 输入应用名称及应用描述。

图 2-16 创建应用



创建应用


* 应用名称 自定义页面

* 应用描述 63

确定 取消

- 步骤4** 单击“确定”。
- 结束

删除应用

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 在应用列表中选择待操作的应用，单击.
- 步骤4** 在弹框中，单击“确定”，完成应用删除。
- 结束

2.4 页面管理

页面管理插件可以管理该应用下的全部页面，可以新增页面，可以新增文件夹，以及对页面或者文件的增删改操作。

假如有一个前端工程：

```
- project
- views
  |_ Index.vue
  - Page2.vue
  - TodoFolder
  |_ Todo.vue
```

设计器的页面概念就相当于上述工程中的一个页面或者文件夹，每个页面有对应的路由，您可以根据路由访问对应的页面。

新增文件夹

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。


- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击，进入“页面管理”页面。
- 步骤5 单击“页面管理”的新增文件夹按钮。
- 步骤6 设置基本属性，如输入文件夹ID及设置路由。

图 2-17 创建文件夹



- 步骤7 单击“保存”，完成文件夹创建。

----结束

删除文件夹

说明

删除文件夹前，请确保文件夹为空，删除文件夹中的页面，具体操作请参考[删除页面](#)。




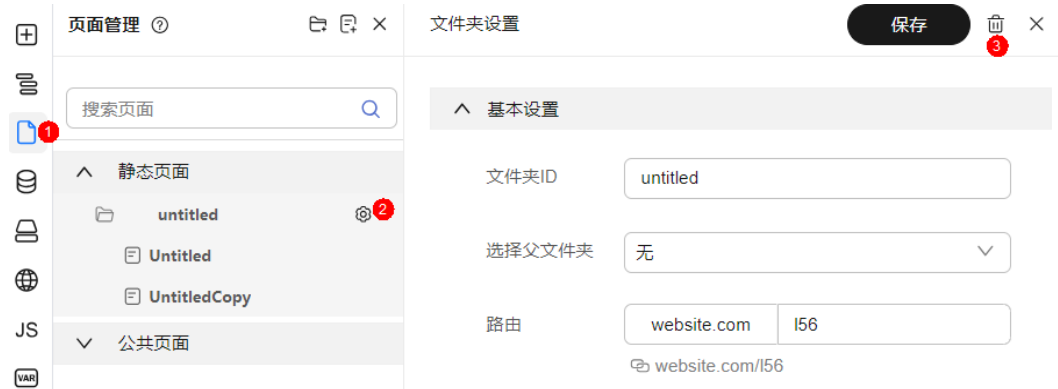
- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击，进入“页面管理”页面。
- 步骤5 单击，进入页面设置页面。
- 步骤6 单击。

图 2-18 删除文件夹



步骤7 在弹框中，单击“确定”，完成文件夹删除。


----结束

新增页面

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

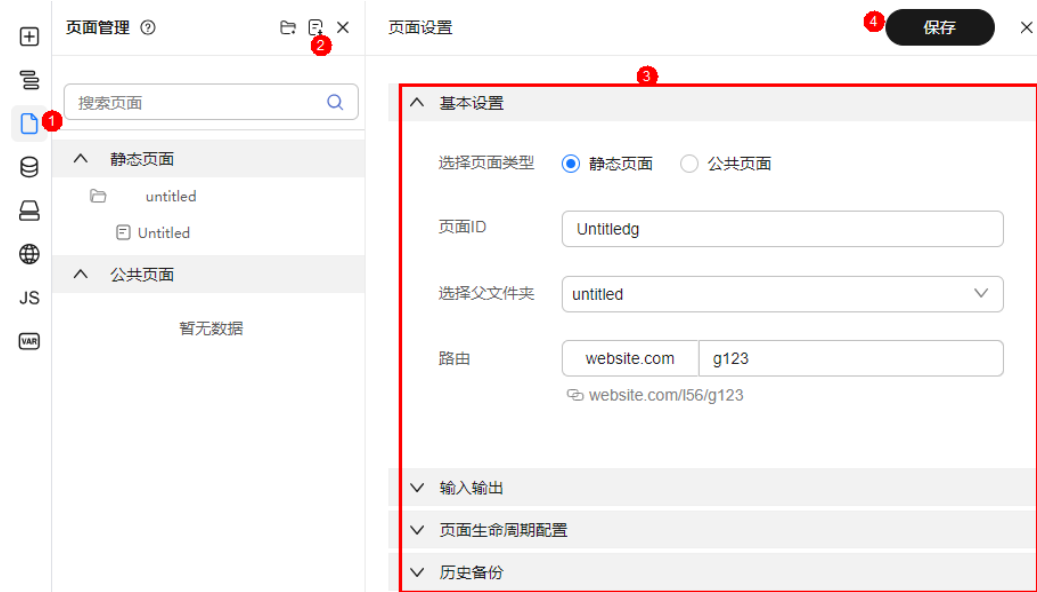
步骤4 在左侧插件栏中，单击，进入“页面管理”页面。

步骤5 单击“页面管理”的新增页面按钮。

步骤6 设置页面基本属性。

- 选择页面类型：可选“静态页面”或“公共页面”。
- 页面名称：只允许包含英文字母，且以大写开头驼峰格式，如DemoPage。
- 选择文件夹：下拉框中选择文件夹名称。
- 路由：输入路由信息，只允许包含英文字母、数字、下划线、中划线和正斜杠组成，且以英文字母开头。

图 2-19 创建页面



步骤7 (可选) 设置输入输出配置。

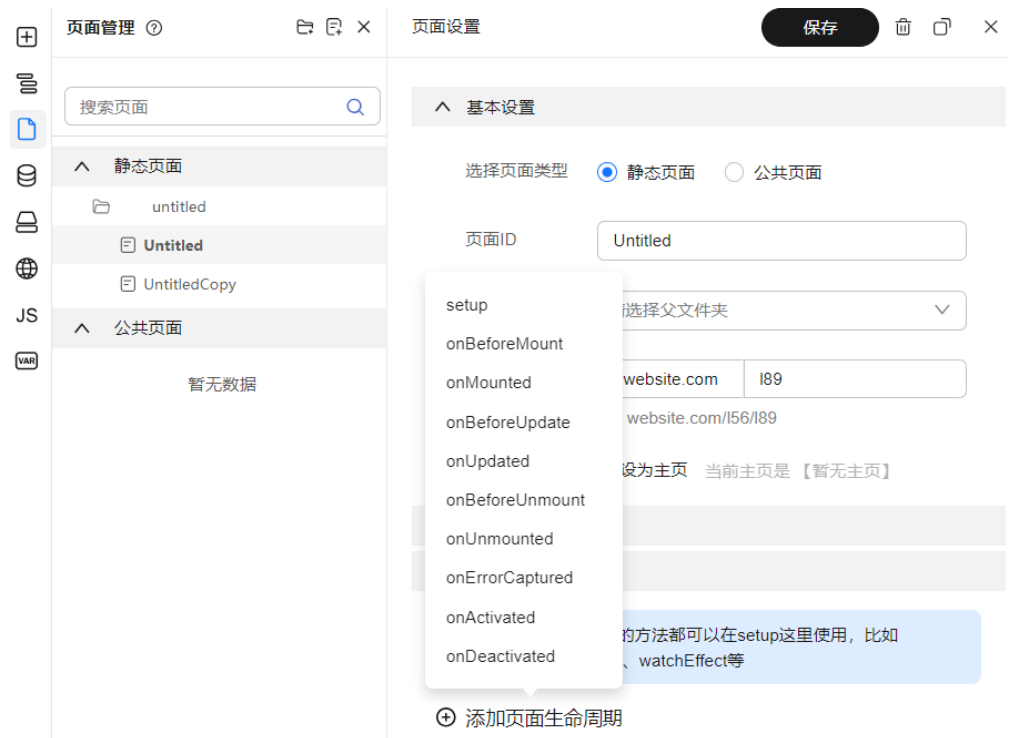
- 输入配置：传递给页面的参数(类似组件的输入属性)。
- 输出配置：页面传递出的事件(类似组件触发的输出事件)。
- 根元素设置：开启后设置根元素为Body，默认为div。

步骤8 (可选) 页面生命周期配置。

1. 单击“添加页面生命周期”。
2. 选择生命周期函数，例如onMounted、setUp、onUpdated等。

周期函数详细说明可参考[生命周期选项](#)。

图 2-20 添加页面生命周期



3. 编写生命周期函数，单击“确定”。

图 2-21 编写生命周期函数



步骤9 单击“保存”。

步骤10 在弹框中输入历史备份信息，单击“确定”，完成文件夹创建。

----结束

复制页面


步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，进入“页面管理”页面。

步骤5 鼠标悬浮在待复制的页面名称上，将显示设置按钮。

步骤6 单击，进入页面设置页面。

步骤7 单击，修改页面ID及路由配置。

图 2-22 设置页面基本信息



步骤8 单击“保存”。

步骤9 在弹框中输入历史备份信息，单击“确定”，完成页面复制。


----结束

还原页面历史记录


步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，进入“页面管理”页面。

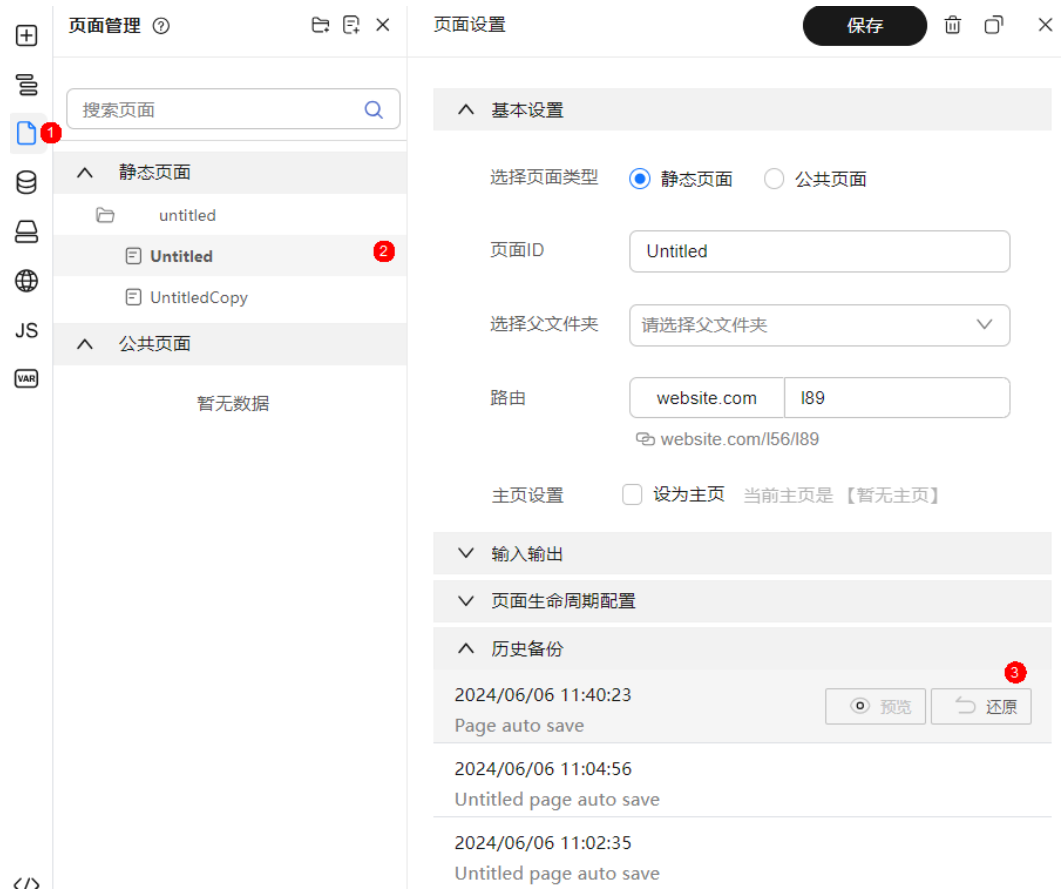
步骤5 鼠标悬浮在待复制的页面名称上，将显示设置按钮。

步骤6 单击，进入页面设置页面。

步骤7 鼠标悬浮在待恢复的历史备份记录上，将显示操作按钮。

步骤8 单击“还原”。

图 2-23 还原页面



步骤9 在弹框中，单击“确定”，完成页面历史记录还原。

----结束

删除页面


步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，进入“页面管理”页面。

步骤5 鼠标悬浮在待复制的页面名称上，将显示设置按钮。

步骤6 单击，进入页面设置页面。


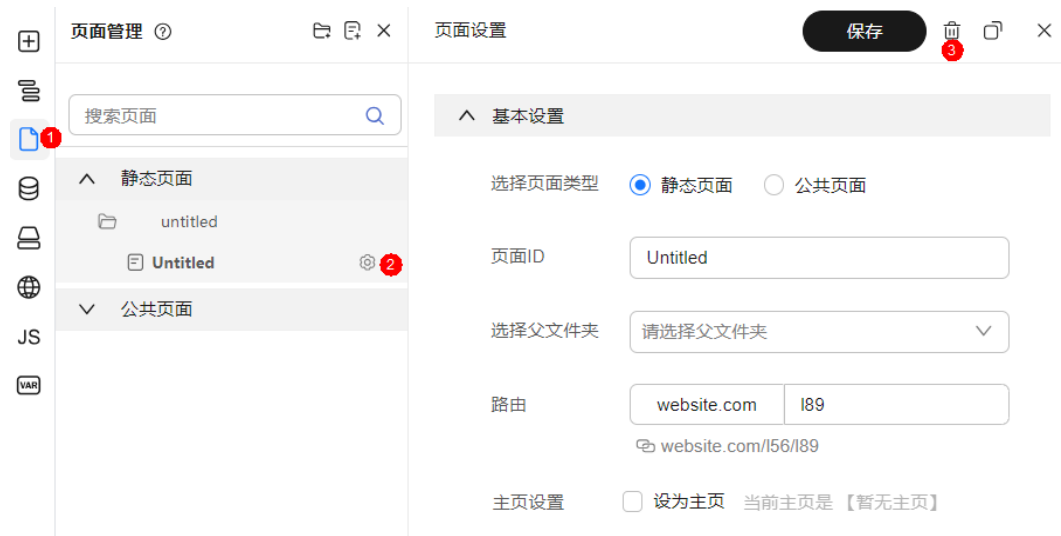
步骤7 单击。

图 2-24 删除页面



步骤8 在弹框中，单击“确定”，完成页面删除。

----结束


设置主页

主页就是整个应用的home页面，我们可以在页面管理中配置主页（即path为“/”时渲染的页面）。


步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

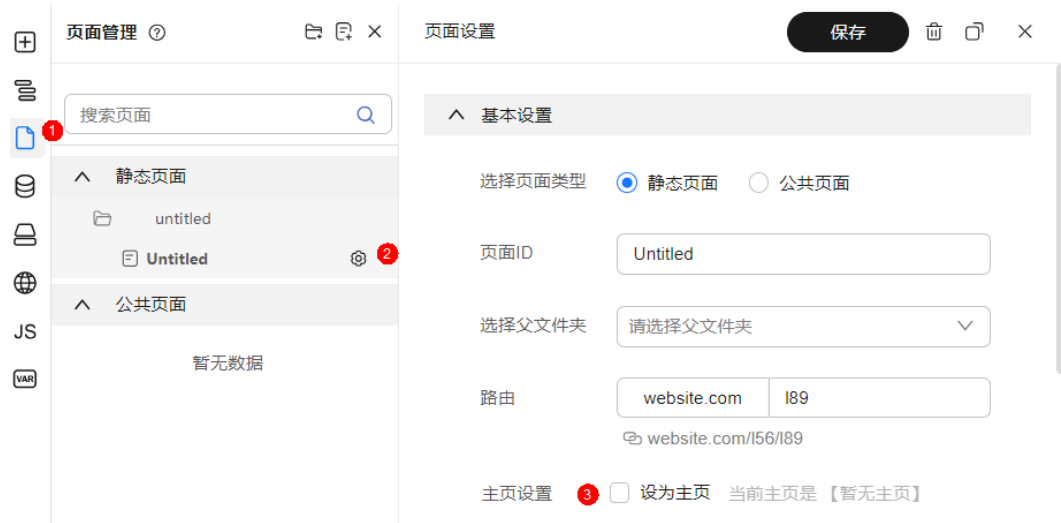
步骤4 在左侧插件栏中，单击，进入“页面管理”页面。

步骤5 鼠标悬浮在待复制的页面名称上，将显示设置按钮。

步骤6 单击，进入页面设置页面。

步骤7 在基本设置页签下，勾选“设为主页”。

图 2-25 设置主页



步骤8 在弹框中，单击“确定”，完成主页设置。

----结束

2.5 使用组件

概述

组件是低代码开发的构建块，就像积木一样。它们是可重复使用的小部件，可以轻松搭建和管理网页。举个例子，按钮、导航栏和表单都可以是组件。这有助于我们快速搭建网站。

下面介绍如何在低代码中使用组件，实现页面快速搭建。

从物料面板中添加组件


- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击，展开物料资产包。
- 步骤5 在物料资产包中选择组件，并拖拽至中心画布中。

图 2-26 添加组件

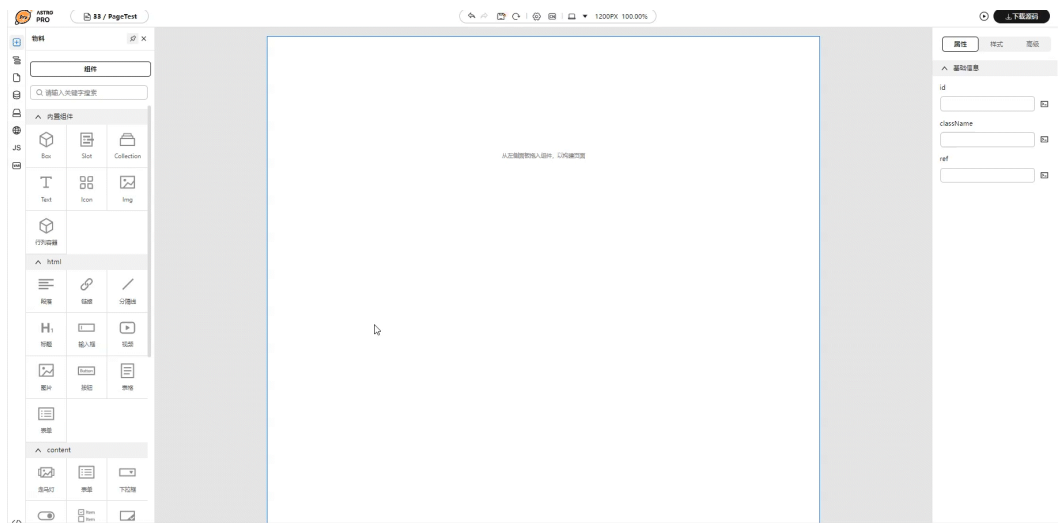
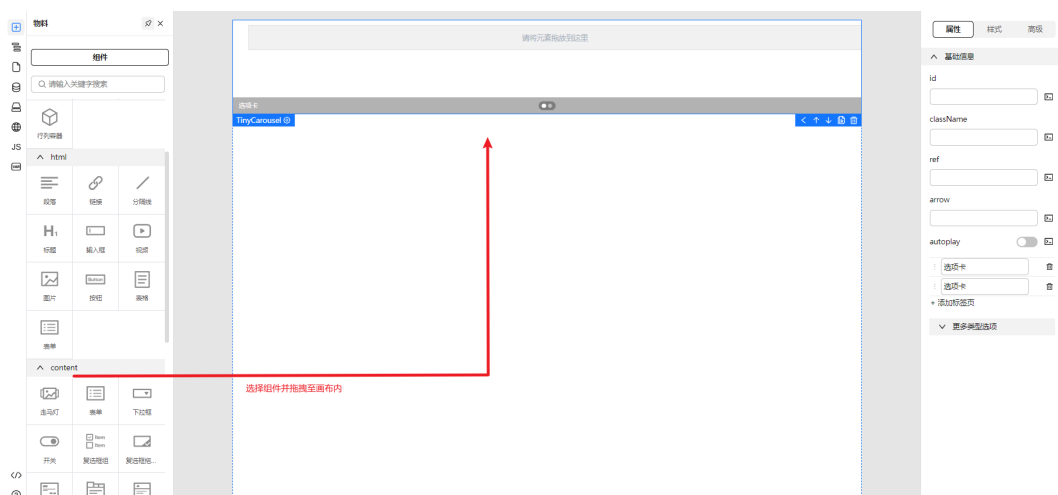


图 2-27 添加组件



----结束

从画布中直接添加组件

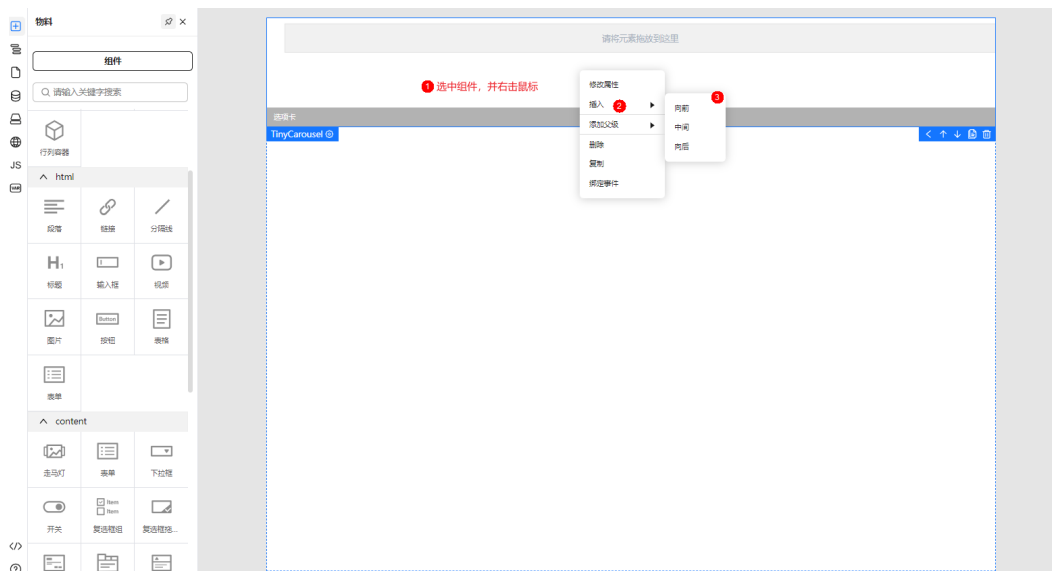
对于复杂的页面，嵌套层级可能很深，直接拖拽并不能很好的拖入到指定的层级中，所以，设计器提供在画布中右键精准添加组件的能力。

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，然后单击鼠标右键。
- 步骤5** 选择“插入”，可以向前、向后、向中间位置进行插入（相对于当前选中组件而言）。

图 2-28 插入组件

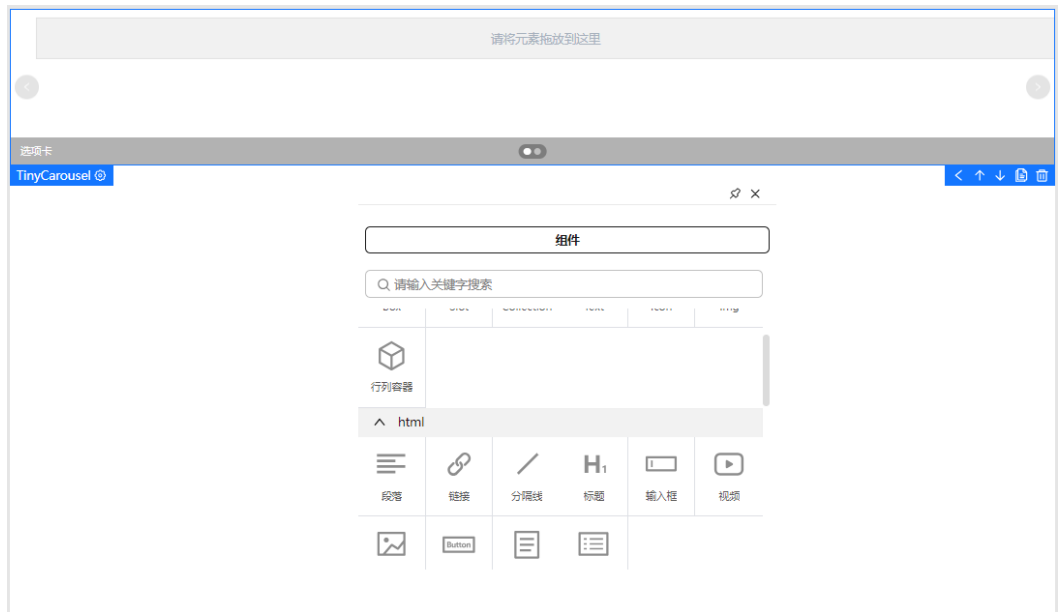


图 2-29 选择插入位置



步骤6 弹框中选择待插入的组件，即可插入指定位置。

图 2-30 选择组件



----结束

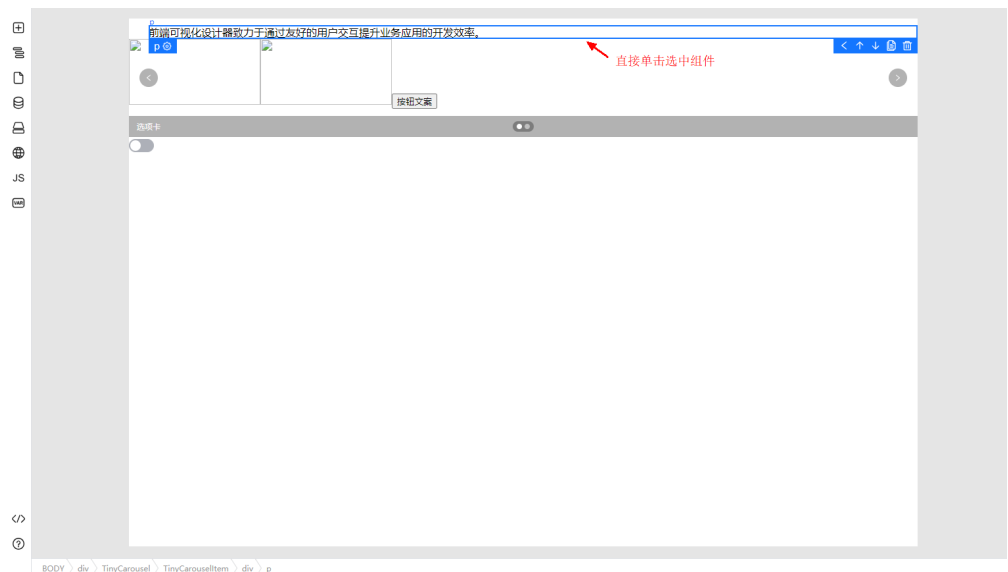
选中组件

添加了组件之后，您可以选中组件，对组件进行修改、删除、移动等操作。

以下为选中组件的相关方法：

- 从画布中直接单击组件选中组件。

图 2-31 直接单击选中组件



- 在左侧大纲树插件中，以树的形式展示了当前页面中所有的组件，单击树节点选中画布中的组件。

图 2-32 单击树节点选中组件



- 底部节点树展示了从根节点到当前选中组件的层级节点，单击底部的节点树的某一个节点，选中父组件。

图 2-33 展示当前选中组件的层级节点

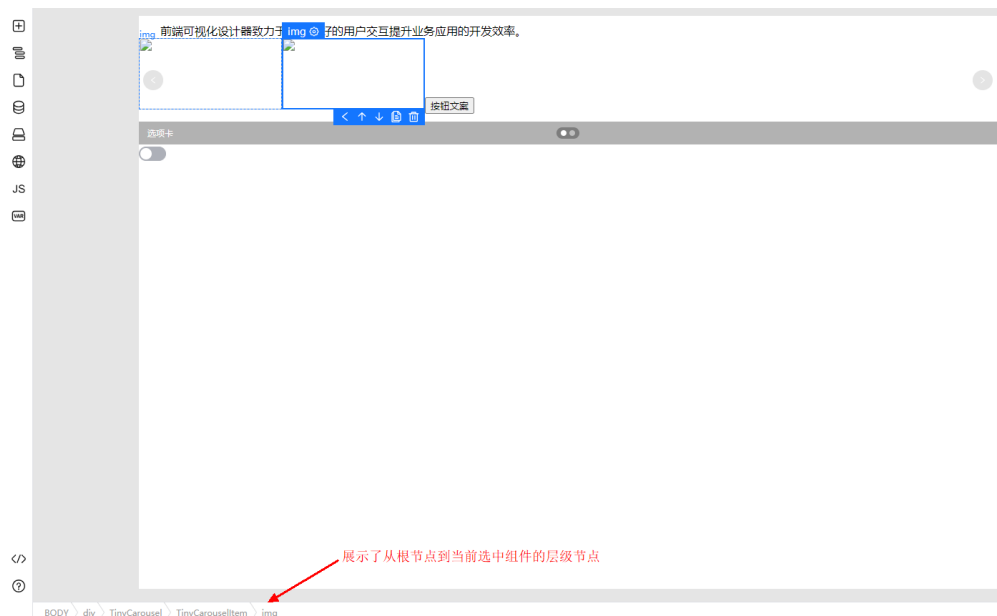
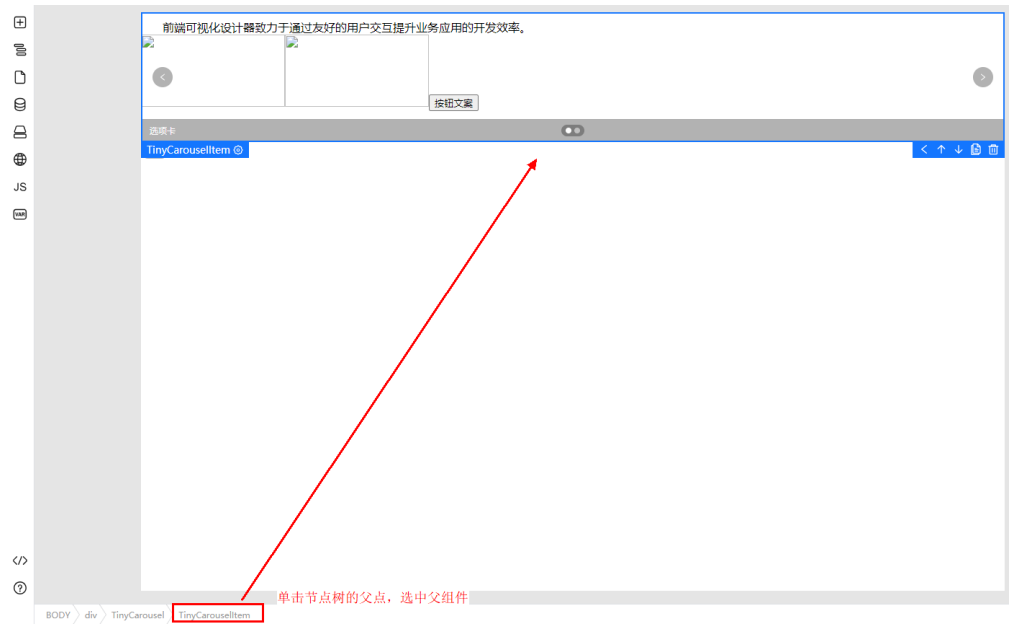


图 2-34 单击节点树的父点，选中父组件



编辑组件

选中了组件之后，我们还可以设置组件的**属性**、**样式**和**绑定事件**，同时可以对组件进行上移、下移、复制和删除等操作。

图 2-35 组件上移

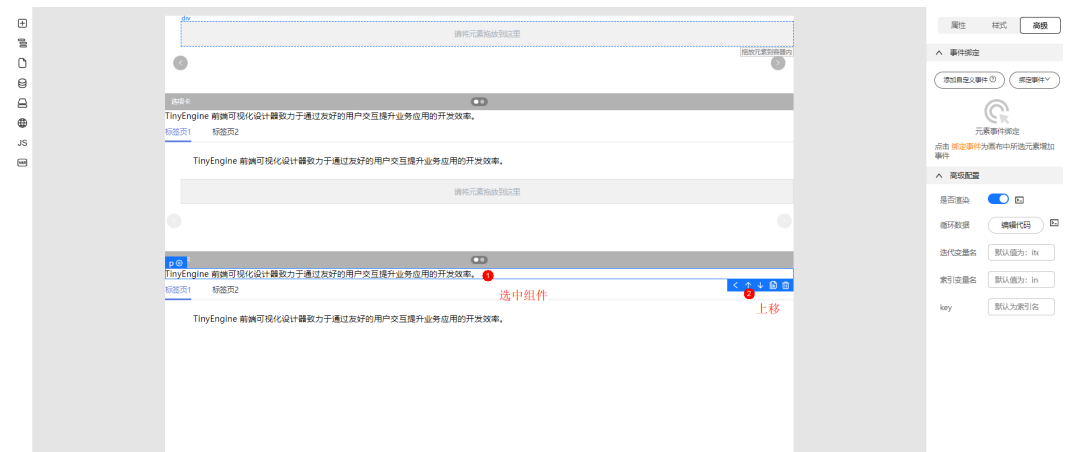


图 2-36 组件下移

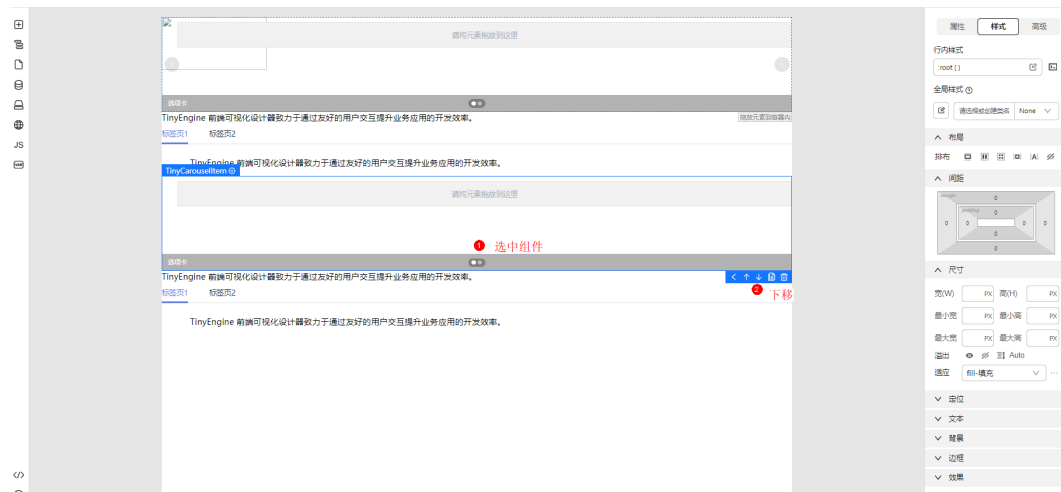
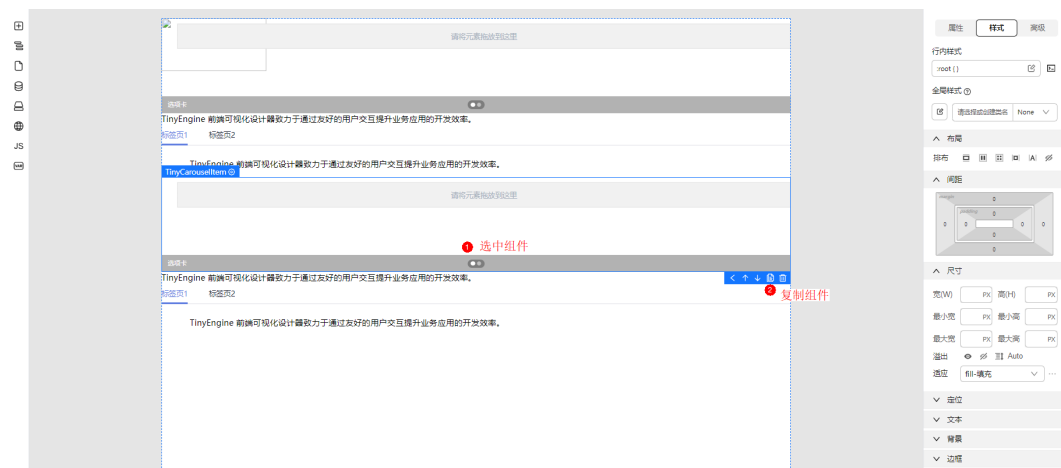


图 2-37 复制组件



2.6 属性设置

使用说明

添加组件后您可以通过右侧的属性设置面板，对当前选中组件进行属性的设置。

参数说明

通用参数：

- id: 规定HTML元素的唯一的id。
- className: 用于属性定义元素的类名；通常用于指向样式表的类和JavaScript中。
- ref: 接受一个内部值，返回一个响应式的、可更改的ref对象，此对象只有一个指向其内部值的属性。

更多参数说明详情请参考[vuejs](#)。

2.7 样式设置

2.7.1 通过样式面板配置样式

使用说明

添加组件后您可以通过右侧的属性设置面板，对当前选中组件进行样式的设置。

通过样式面板配置样式

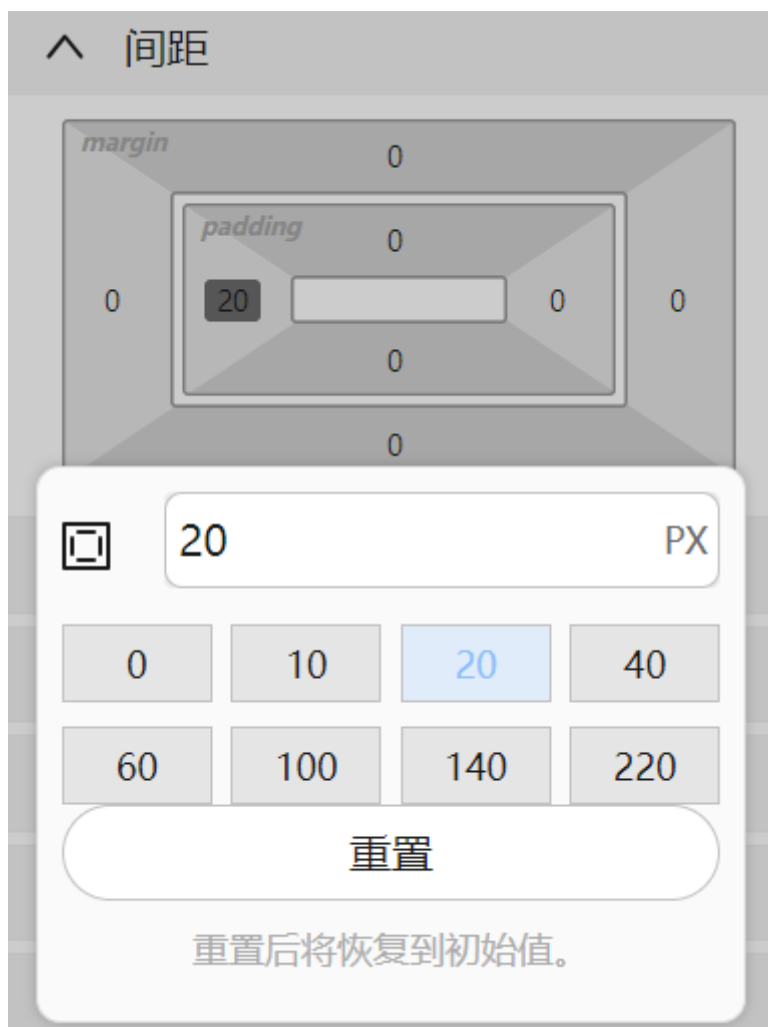
- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5 在样式面板中设置组件样式。
 - 布局：支持“block-块级布局”、“flex-弹性布局”、“grid-网格布局”、“inline-block-内联块级”、“inline-内联”和“eye-invisible-隐藏”六种形式，请按需选择。

图 2-38 设置布局



- 间距：默认间距为0，可单击待设置的间距数值进行自定义修改。

图 2-39 设置间距



- 尺寸：设计组件的宽高，溢出处理策略及适应方式。

图 2-40 设置尺寸



- 定位：设置组件位置和大小如何计算方式。
 - 默认定位：没有特别的定位，组件的位置会按照正常的文档流进行布局。
 - 相对定位：组件的位置相对于它在文档流中的位置进行偏移。组件仍然占据原来的空间，不会影响其他元素的布局。
 - 绝对定位：组件的位置相对于最近的已定位祖先元素(如果没有已定位的祖先元素，则相对于初始包含块)进行偏移。组件不占据原来的空间，会从文档流中删除，并影响其他元素的布局。
 - 固定定位：组件的位置相对于浏览器窗口进行固定定位。组件不占据原来的空间，会从文档流中删除，并影响其他元素的布局。
 - 粘性定位：组件在滚动时“粘”在某个位置，直到达到某个条件才会离开这个位置。粘性定位可以让页面在滚动时保持某些组件的可见性。

图 2-41 设置定位



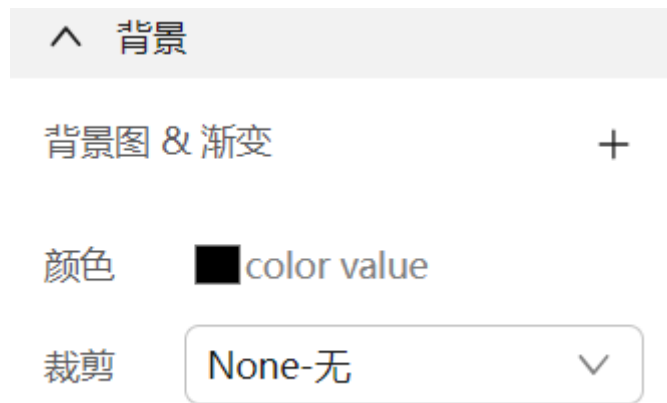
- 文本：设置文本的字号、字体、行高、字重、颜色、对齐方式、风格及修饰。

图 2-42 设置文本



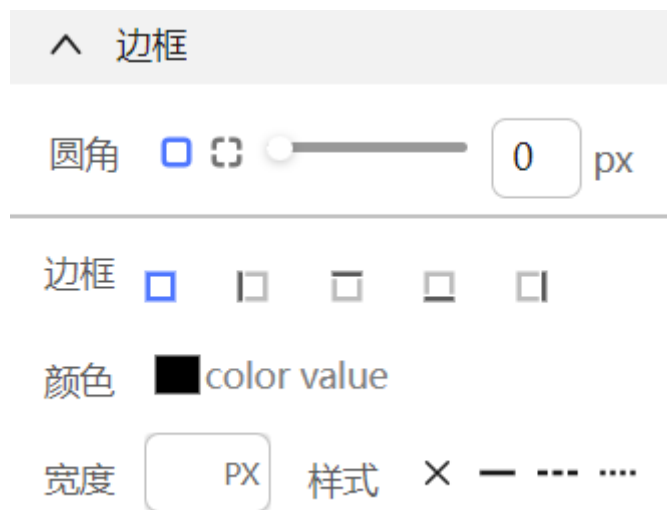
- 背景：设置组件背景，背景图、渐变类型、颜色及裁剪方式。

图 2-43 设置背景



- 边框：设置边框的位置、颜色、位置宽度以及是否为圆角。

图 2-44 设置边框



- 效果：设置组件的透明度、轮廓样式及光标样式。

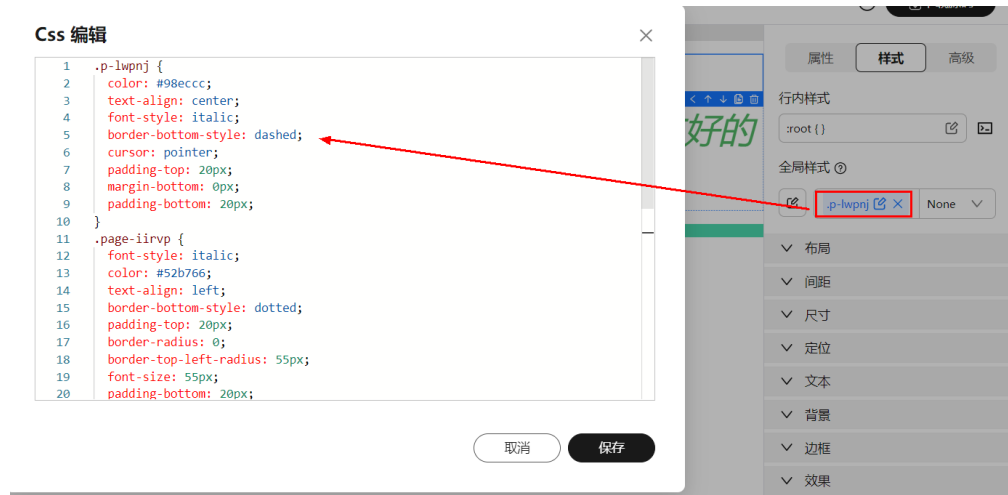
图 2-45 设置显示效果



步骤6 样式设置完成后，设计器会自动完成以下动作。

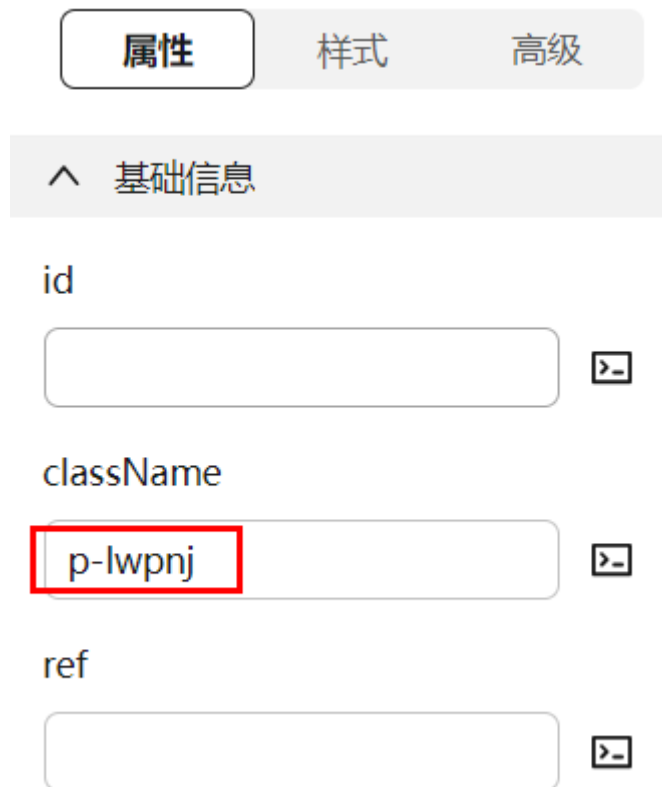
1. 如果当前样式在选择器中没有选中任何一个类名或者ID，会自动生成随机类名。
2. 将在样式面板设置完成的样式写入类名规则中。

图 2-46 样式写入到类名样式



3. 将样式选择器自动生成或者选中的类名绑定到当前组件属性中。

图 2-47 属性设置面板



----结束

2.7.2 直接编写样式代码

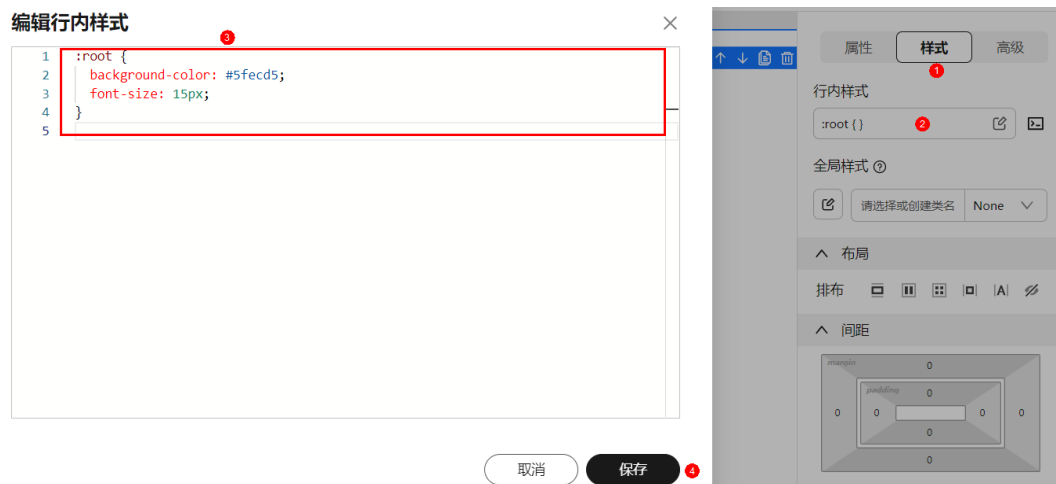
样式面板提供的直接配置的样式满足了大部分的基础样式需求，但是，如果还是不能满足我们的需求，我们还提供了直接编写样式代码的方式来配置样式。

编写行内样式

等效于直接在html标签style属性里直接声明样式，权重高，用于覆盖样式。

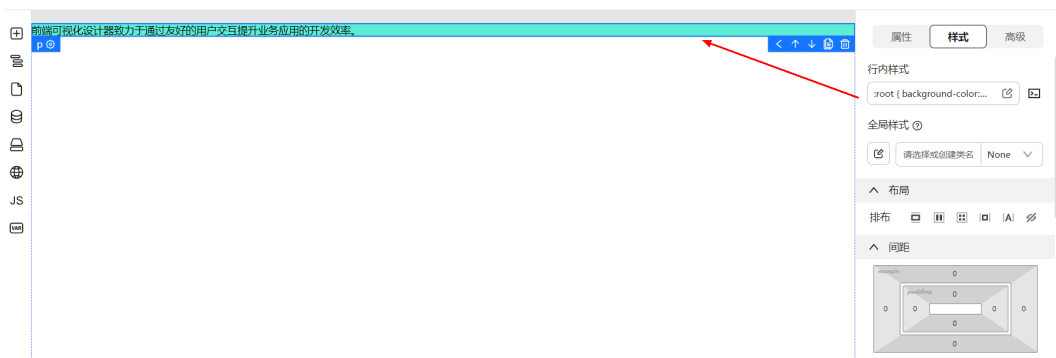
- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5** 单击行内样式的编辑框。
- 步骤6** 在弹框中直接编写行内样式。

图 2-48 编写行内样式




- 步骤7** 单击“保存”，完成行内样式编辑。

图 2-49 效果展示



----结束

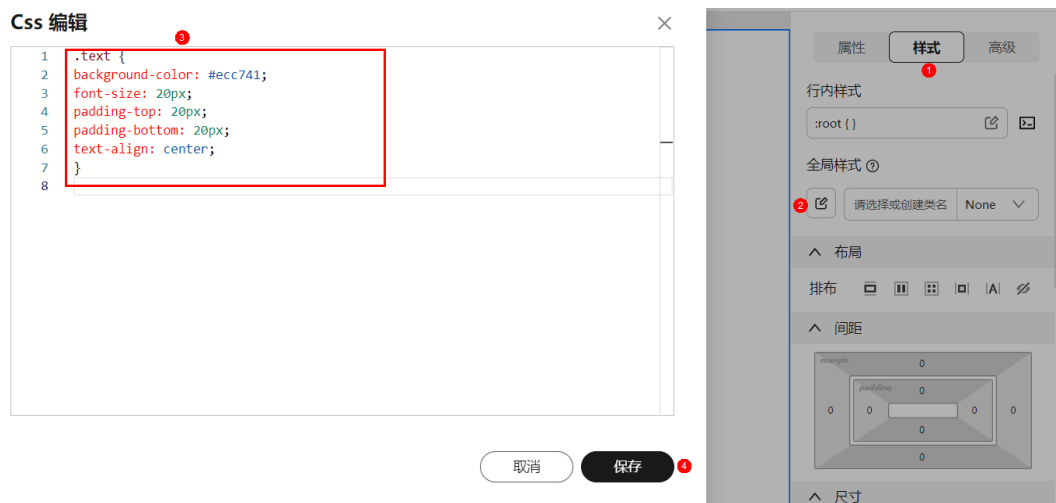
编写全局样式

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5** 单击全局样式的 。
- 步骤6** 在弹框中直接编写全局样式。

说明

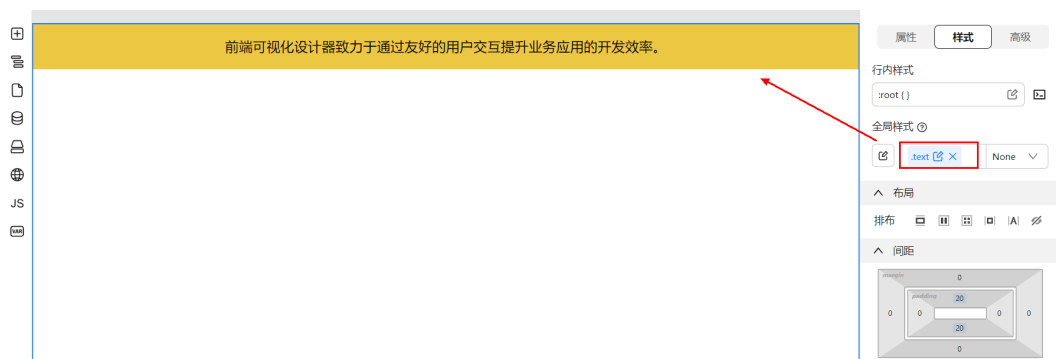
可以直接声明多条CSS样式或其他媒体查询的样式。

图 2-50 编写全局样式



- 步骤7** 单击“保存”，完成全局样式编辑。
- 步骤8** 通过组件的class或者id进行绑定。

图 2-51 效果展示



----结束

2.7.3 类名管理

2.7.3.1 新增类名

使用说明

当选中一个组件，还没有自动生成类名时，您可以单击输入框手动输入类名，然后继续编辑样式面板的样式。

此时，组件会自动绑定您输入的类名，然后将在样式面板编辑的样式生成代码，写入到全局样式中。

操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。

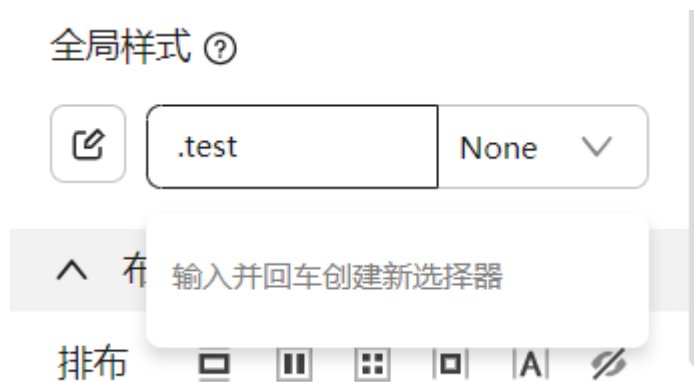
步骤4 画布中选中组件，在组件属性设置面板选择“样式”。

步骤5 单击全局样式的输入框，输入类名，例如.test。

📖 说明

- 类名不能以数字开头。
- 绑定组件属性元素ID：使用格式 #elementID。
- 绑定组件样式类：使用格式 .className。

图 2-52 创建类名



步骤6 单击回车键，类名创建完成。

----结束

2.7.3.2 修改类名

使用说明

已创建的类名支持修改。

说明

类名修改，需注意以下事项：

- 当前组件绑定的类名会被修改。
- 全局样式中同类名会被修改成新类名。
- 其他组件绑定的同类名不会被修改。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5** 单击全局样式选择框的类名或者单击类名旁边的修改按钮，重新输入类名，例如.test123。
- 步骤6** 单击回车键，类名修改完成。

----结束

2.7.3.3 删除类名

使用说明


当类名不在使用，可进行删除操作。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待操作应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5** 单击全局样式选择框类名旁的删除按钮。

删除后当前组件属性不再绑定该类名。

说明

全局样式面板仍保留该类名及样式，其他组件仍在使用该类名。如需删除CSS中的该样式信息，可通过单击全局样式的，进行删除。

----结束

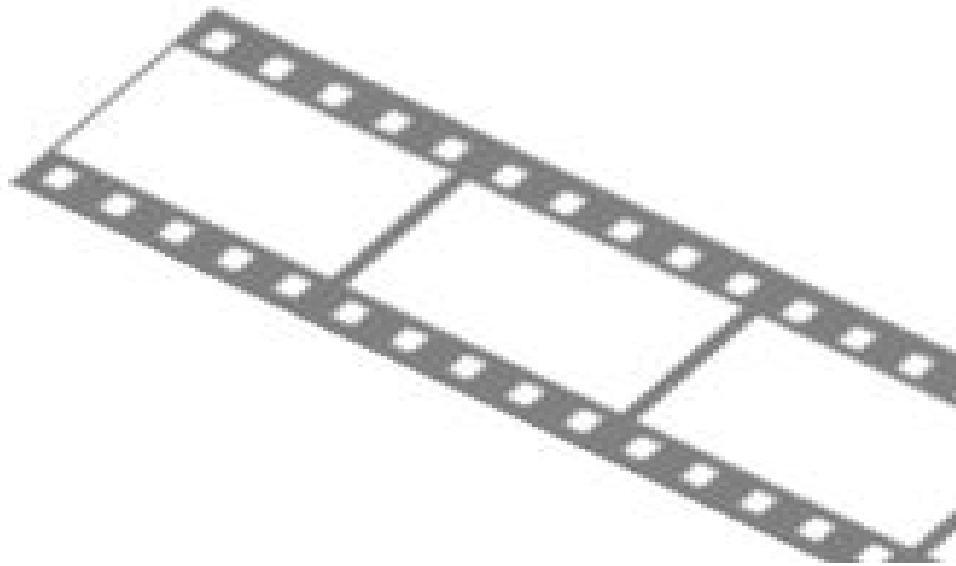
2.7.3.4 选择已有类名

使用说明

除了自动生成和手动新增类名之外，我们也可以通过输入框的下拉框来选择已有的类名。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5** 在全局样式选择框中选择已有的类名，例如 .test1、.test2、.test3等。
通过下拉框选择切换类名，组件将切换至类名绑定的对应样式。



---结束

2.7.4 使用状态选择器

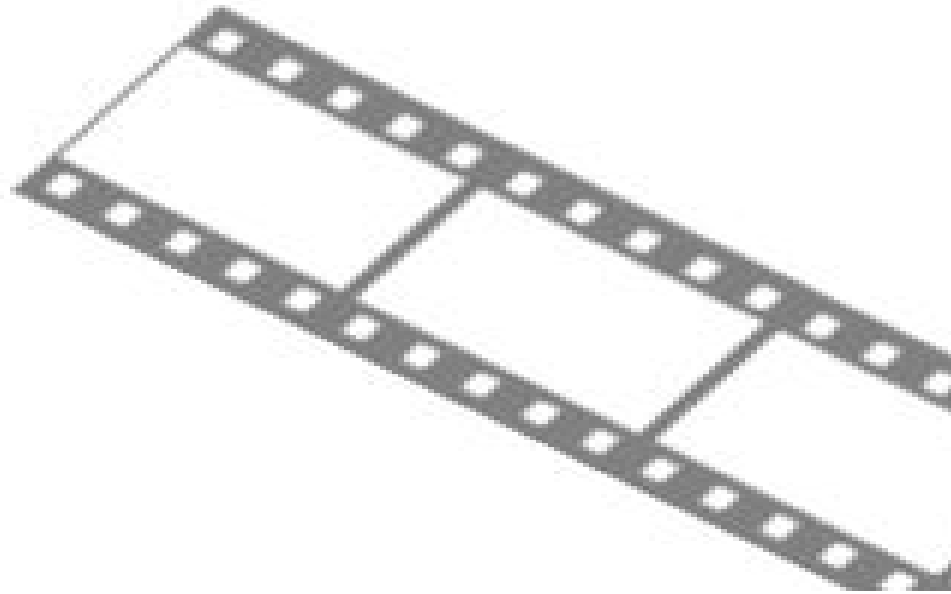
使用说明

在样式选择器右侧，我们还提供了一个状态选择器的下拉框。可以选择hover、focused、pressed、disabled等状态。选择之后，我们此时编辑样式面板，就是相当于在编辑该类名对应状态的样式。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤5** 在全局样式选择框中选择已有的类名或创建新类名，例如 .test，并设置全局样式，此时状态选择器默认为None。
- 步骤6** 在状态选择器下拉框中选择状态，例如hover，并设置全局样式。
从全局样式面板中可以看到，等效于直接编辑一下样式：

```
.test:hover{  
  /** 这里会注入样式面板编辑的样式 */  
}
```
- 步骤7** 查看CSS编辑页面，可见已存在 .test和 .test:hover两条样式记录。
- 步骤8** 鼠标悬浮在组件上查看样式切换效果。



----结束

2.7.5 行内绑定状态变量

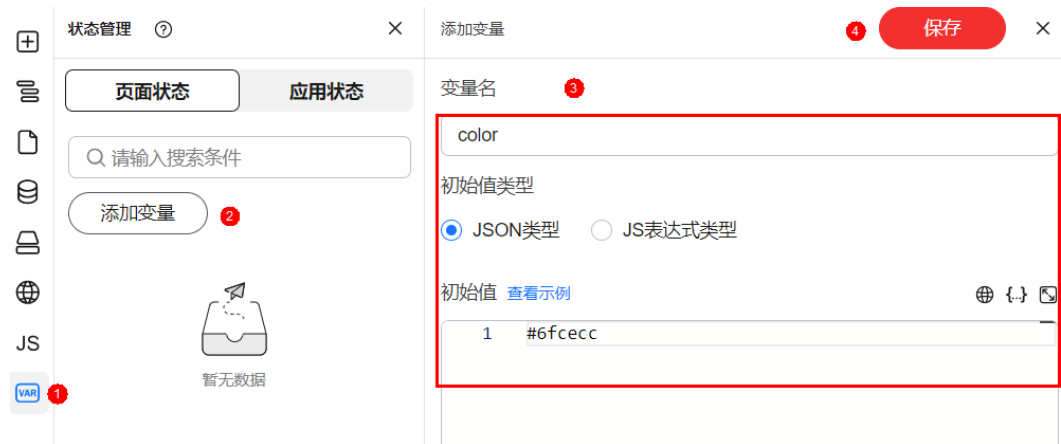
使用说明

设计器提供了一种功能，允许使用定义变量来动态计算组件的位置。这些计算结果随后可以直接应用到组件的行内样式style属性中，从而实现动态样式的调整和布局的优化。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 添加变量，例如：bgcolor，用来控制背景颜色，具体操作请参考[添加页面变量](#)。

图 2-53 添加变量




- 步骤5** 画布中选中组件，在组件属性设置面板选择“样式”。
- 步骤6** 单击行内样式的编辑框后的 。
- 步骤7** 在弹框中进行变量绑定。

图 2-54 绑定变量



- 步骤8** 单击“确定”，变量绑定完成。
- 步骤9** 通过修改变量，查看组件样式展示效果。

图 2-55 修改前样式

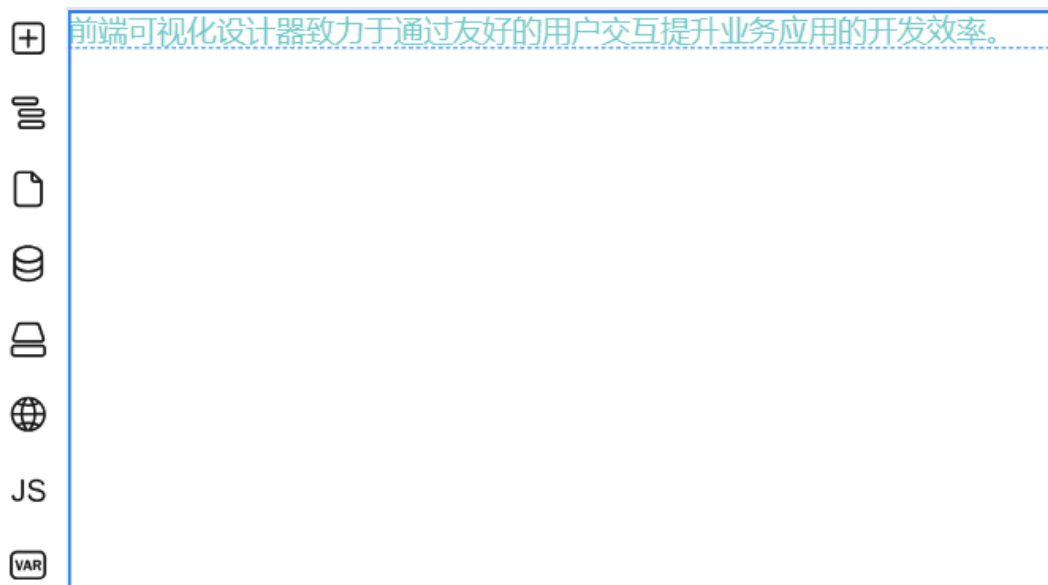


图 2-56 修改变量

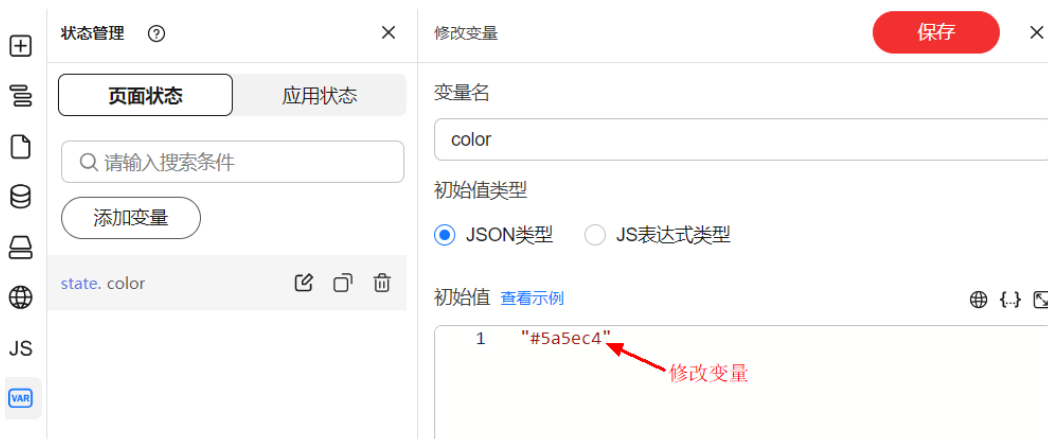
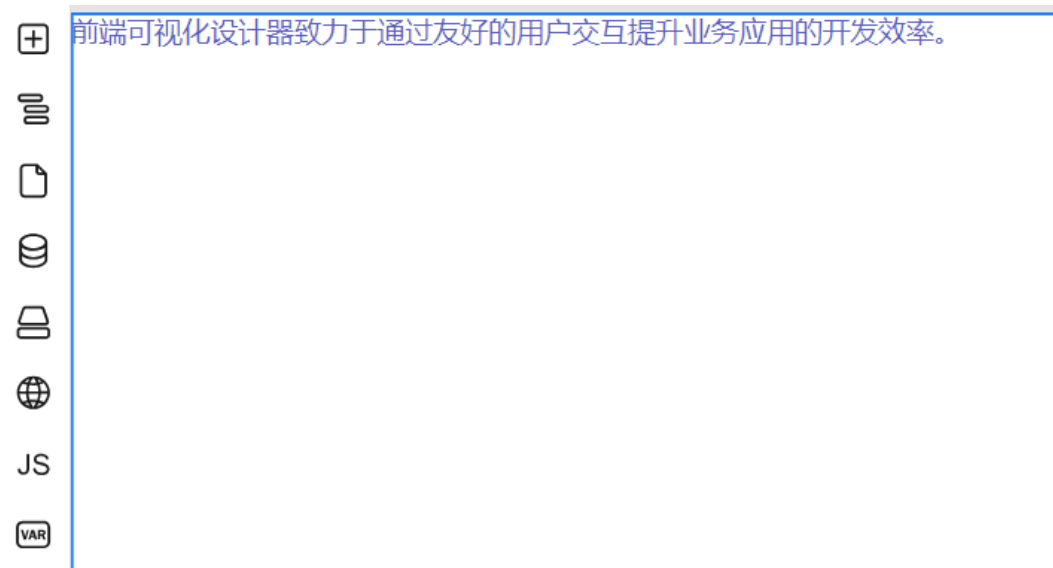


图 2-57 修改后样式

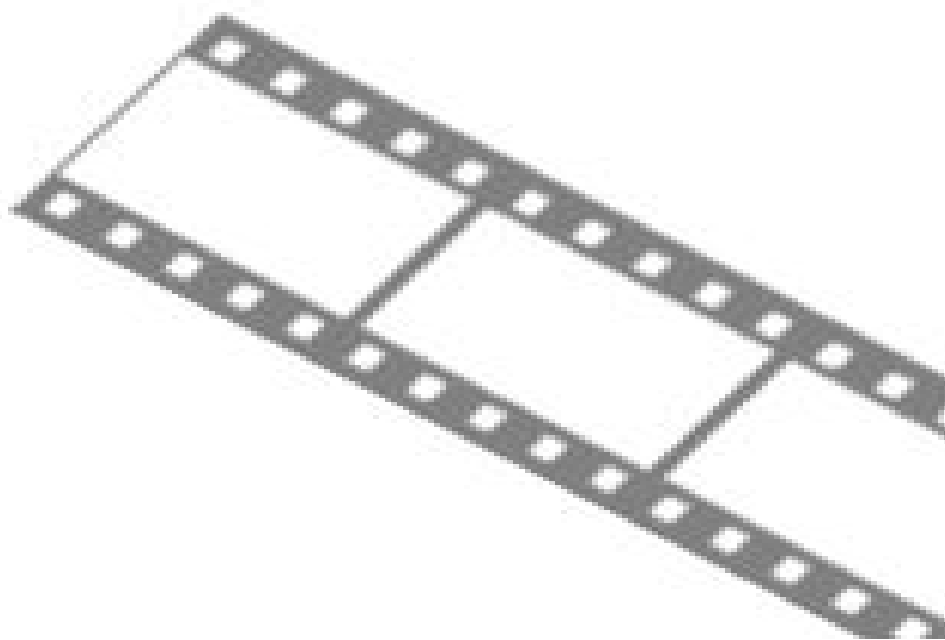


----结束

2.8 高级设置

2.8.1 条件渲染

在页面开发中，可能需要根据某些条件来动态显示或隐藏页面中的内容，例如：如果您希望当用户已经登录的时候，显示“欢迎登录”的文字，未登录的时候，显示“请登录”的文字。



- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4 拖拽组件至画布，分别输入希望展示的文字。
- 步骤5 添加变量，例如state.isLogin，具体操作可参考[添加页面变量](#)。

图 2-58 添加变量




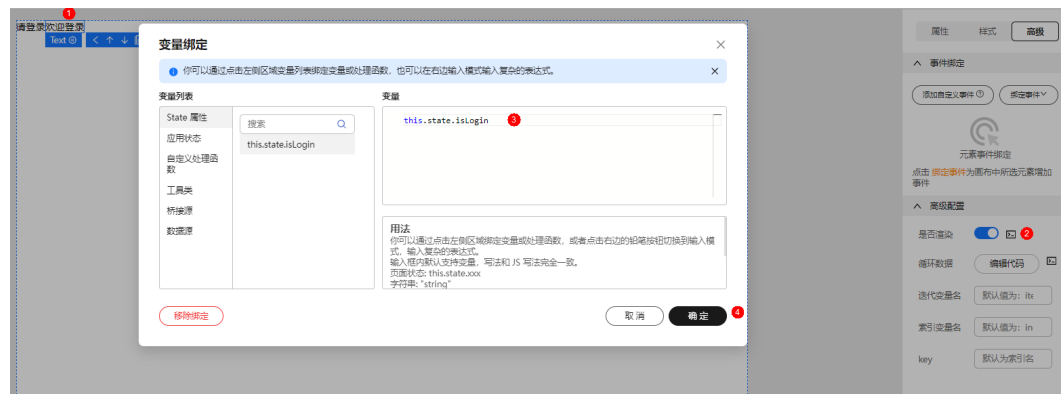
- 步骤6 选中组件，在组件属性设置面板选择“高级”。
- 步骤7 单击“是否渲染”后的，进行变量绑定。

图 2-59 绑定变量



- 步骤8 选项绑定的变量，单击“确定”。
- 绑定成功后可根据变量state.isLogin的值，查看渲染效果。

图 2-60 state.isLogin 为 false 时

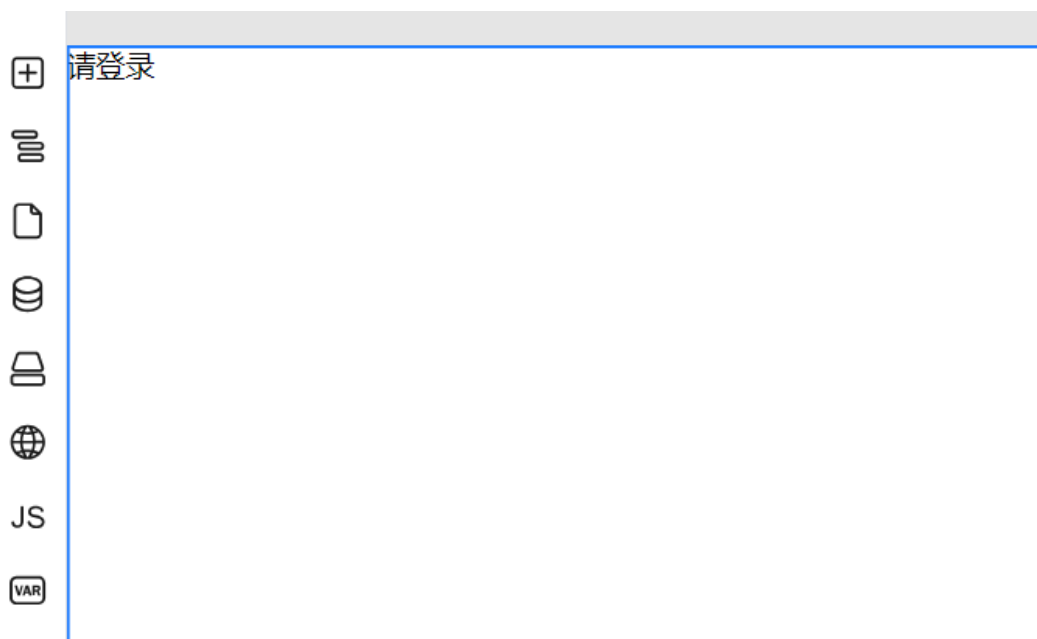
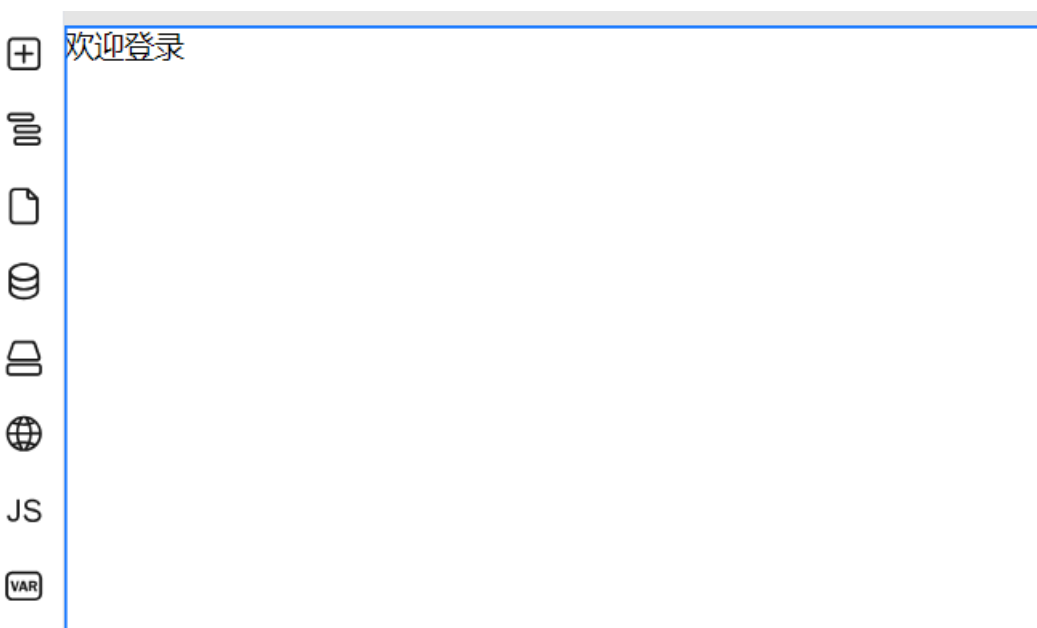


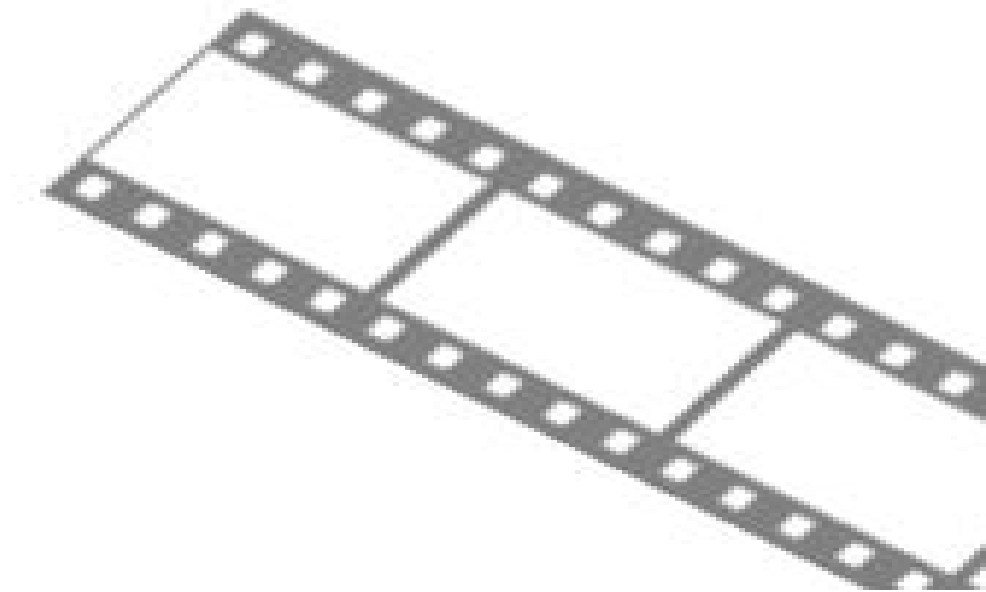
图 2-61 state.isLogin 为 ture 时



----结束

2.8.2 循环渲染

页面可能有若干份重复的、动态生成的内容，例如商品列表页、表格数据。这时候需要用到循环渲染。您可以在高级面板中指定循环数据绑定的变量、迭代的变量名、索引变量名、以及唯一的key。



- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4 拖拽组件至画布，例如拖拽一个“段落”组件。
- 步骤5 添加变量，例如loop.isLogin，具体操作可参考[添加页面变量](#)。

图 2-62 添加变量




- 步骤6 选中组件，在组件属性设置面板选择“高级”。
- 步骤7 单击“循环数据”后的 ，进行变量绑定。
- 步骤8 选择绑定的变量，单击“确定”。

图 2-63 绑定变量

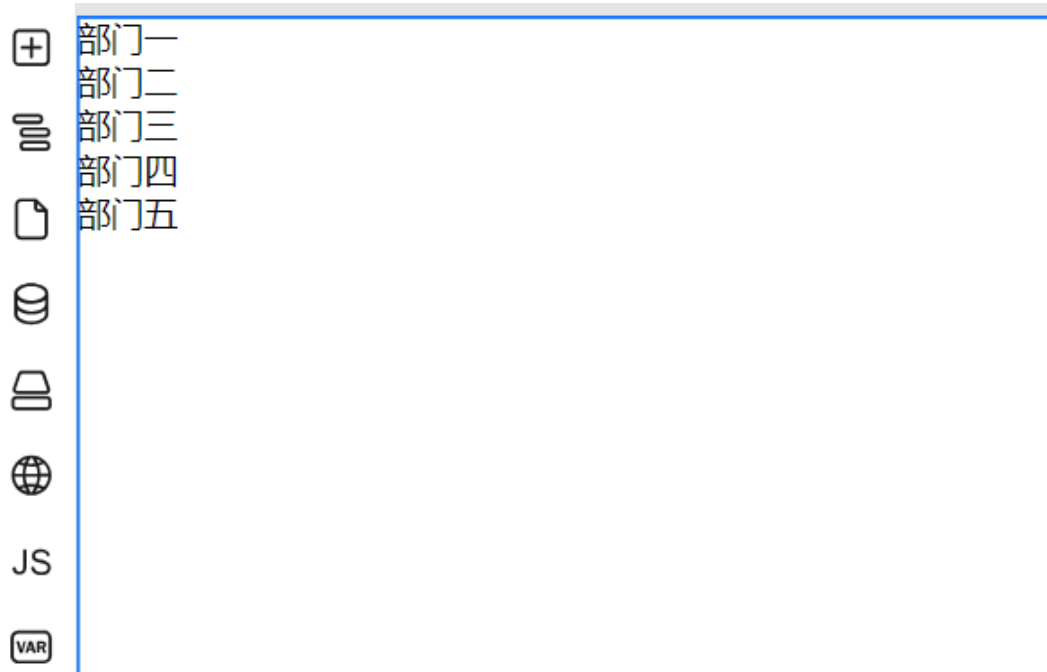


步骤9 为展示内容添加变量，默认为item。
绑定成功后，查看渲染效果。

图 2-64 添加变量



图 2-65 查看渲染效果



----结束

相关概念关联：

- 循环数据，即需要循环渲染的数组，在这里是state.imageList。
- 迭代变量名，在循环渲染子项对应的变量名，默认为item。
- 索引变量名，循环渲染的索引变量名，默认为index。
- key，标识唯一的key，默认为index。

最终出码：

```
<template>  
<div v-for="(item, index) in state.imageList" :key="index">  
<span>{{ item.title }}</span>  
<!--列表细节-->  
</div> </template>
```

2.8.3 绑定事件

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。

步骤4 选中组件，在组件属性设置面板选择“高级”。

步骤5 鼠标悬停在“绑定事件”上，将显示事件列表。

步骤6 在事件列表中，单击需要绑定的事件。

步骤7 在弹框中设置绑定事件。

- 方法名称：输入方法名称，名称可以由字母、数字、下划线、\$ 符号组成，不能以数字开头。

说明

可以选择已有方法或者添加新方法，如果选项添加新方法，单击“确定”之后将在JS面板中创建一个该名称的新方法。

- 扩展参数：调用当前事件传入的真实参数，数组格式，追加在原有事件参数之后如: onClickNew(eventArgs, extParam1, extParam2, ...)。

图 2-66 设置绑定事件



步骤8 单击“确定”，弹出JS面板。

步骤9 在JS面板中进行绑定方法的具体逻辑实现。

步骤10 单击“保存”，绑定事件完成。

----结束

2.9 查看大纲树

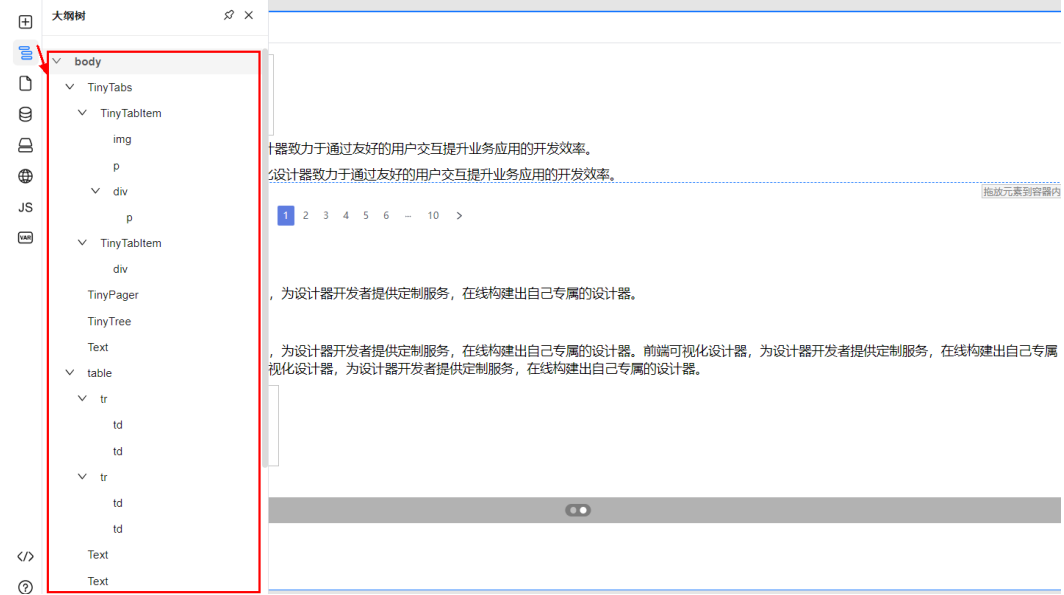
步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击 ，展开并查看页面大纲树。

图 2-67 设置主页



----结束

2.10 数据源管理

2.10.1 获取数据源远程字段

使用说明

您可以通过已有的远程Http接口，快速地生成数据源的字段。

操作步骤



- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击应用模块内的“应用开发”，进入设计器。
- 步骤4** 在左侧插件栏中，单击，展开数据源管理页面。
- 步骤5** 单击，展开设置数据源页面。
- 步骤6** 选择数据源类型（可选对象数组、树结构），并输入数据源名称。
- 步骤7** 单击“获取远程字段”，设置请求地址、请求方式及请求参数。

图 2-68 创建数据源



步骤8 单击“发送请求”，请求成功后获取到接口字段信息，保存后即可生成数据源字段信息。

步骤9 单击“保存”，完成数据源创建。

----结束

2.10.2 添加静态数据

使用说明

您可以直接通过手动创建的方式添加数据源的字段。

操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

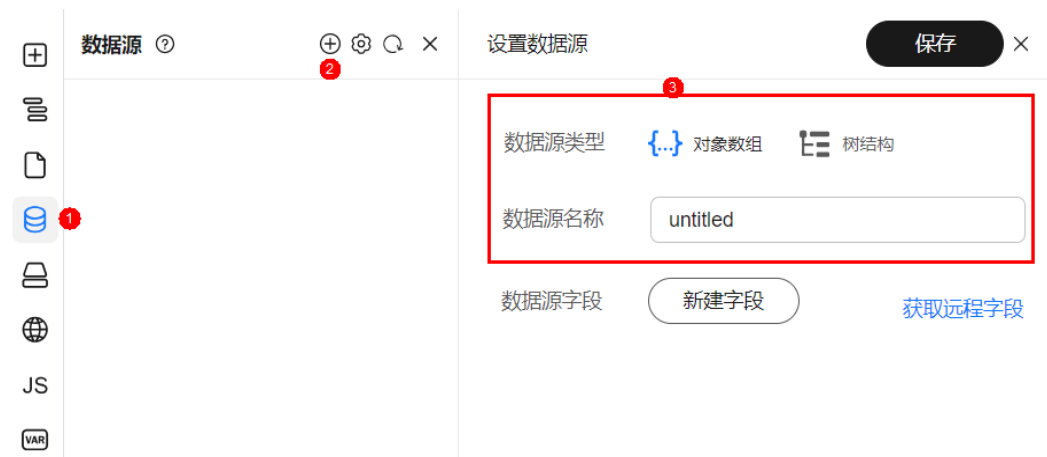
步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，展开数据源管理页面。

步骤5 单击，展开设置数据源页面。

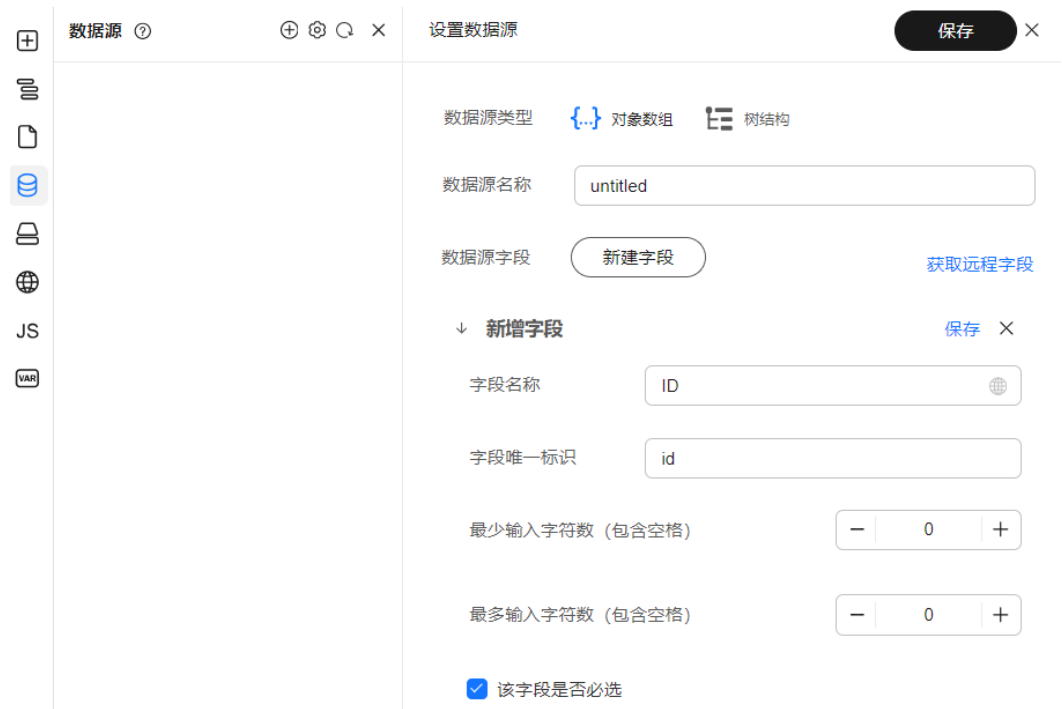
步骤6 选择数据源类型（可选对象数组、树结构），并输入数据源名称。

图 2-69 创建数据源



步骤7 单击“新增字段”，添加静态数据源。

图 2-70 添加静态数据源



步骤8 单击“保存”，完成数据源字段添加。

步骤9 单击“保存”，完成数据源创建。

----结束

2.10.3 使用数据源

使用说明

设计器提供数据源来配合画布上的组件渲染。


本章节以应用于表格组件的表格列为例，为您介绍如何使用数据源。


创建数据源

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，展开数据源管理页面。

步骤5 单击，展开设置数据源页面。

步骤6 配置数据源类型（可选对象数组、树结构），数据源名称以及数据源字段。

图 2-71 创建数据源



图 2-72 新增数据源字段




步骤7 单击“保存”，完成数据源创建。

----结束

绑定数据源

数据源主要载体为Collection组件，因此在使用数据源之前需要先在画布中拖放入Collection组件，然后在属性面板中选择需要绑定的数据源。

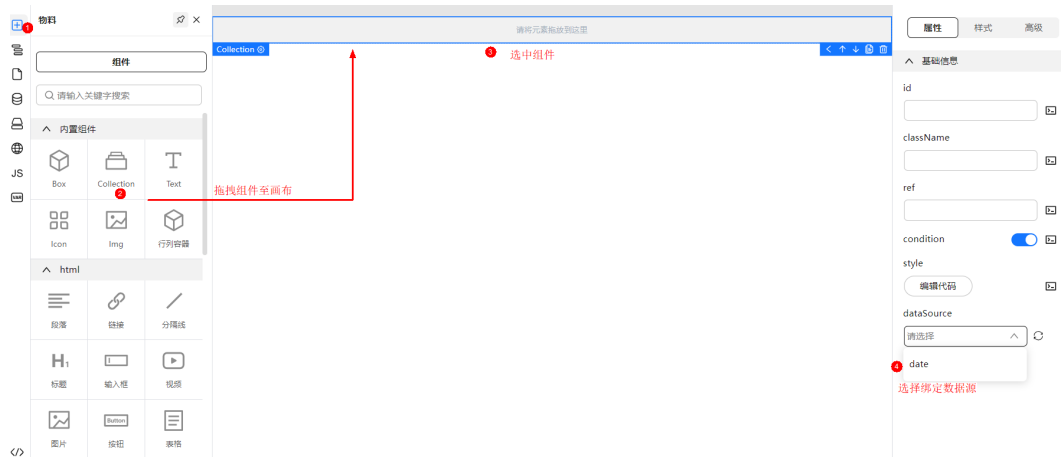
步骤1 在左侧插件栏中，单击，展开物料资产包。

步骤2 在Collection组件，并拖拽至中心画布中。

步骤3 选中组件，在属性设置面板中选择“属性”。

步骤4 dataSource下拉框中选择待绑定的数据源。

图 2-73 绑定数据源



----结束

表格组件中的应用

步骤1 将Grid表格组件拖拽至Collection组件中。

图 2-74 表格拖拽至 Collection 组件



步骤2 单击“确定”，根据提示引入配置数据，自动解析出表格列数据。

图 2-75 解析表格列数据



----结束

2.11 使用工具类方法

2.11.1 添加工具类

在常规代码开发中，我们通常会将一些高频率用到的一些代码片段抽离出来业务代码，使其成为一个公共函数，减少重复的代码，从而达到代码复用的目的。同样，在低代码开发中，我们不可避免的需要编写一些高代码进行组合开发，这里同样存在存储一些公共函数以供各处调用需求。工具类就是在这样的背景之下诞生，我们可以将一些可复用的公共函数编写到工具类中，也可以将一些npm包引用到工具类中，供后续调用。

📖 说明

工具类是应用级别的，即引入之后，即可在该应用下任意页面中进行调用

添加 function 工具类



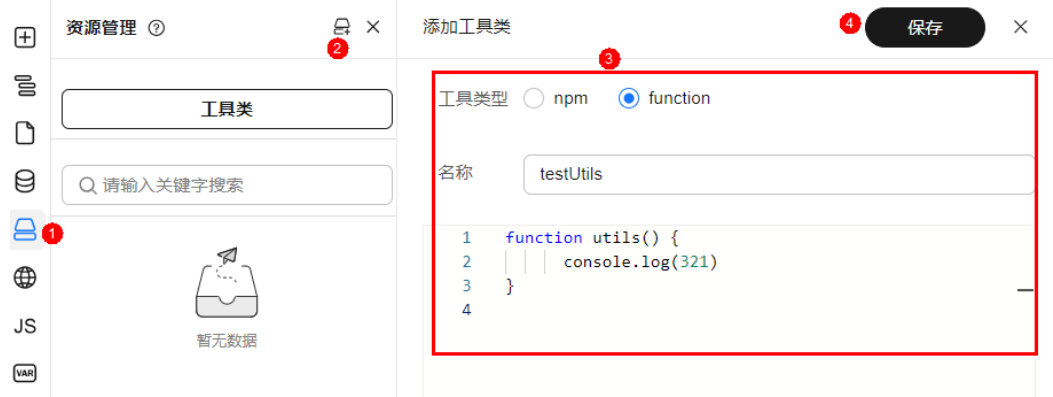
- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击，展开资源管理页面。
- 步骤5 单击，进入添加工具类页面。
- 步骤6 工具类型选择“function”。
- 步骤7 输入工具类的名称，并编写函数代码。

图 2-76 添加 function 工具类



步骤8 单击“保存”，完成function工具类添加。

----结束


添加 npm 工具类

对于一些简单的公共函数来说，直接添加function工具类会很方便，但是对于一些比较复杂的公共函数或者第三方的一些公共函数来说，直接编写函数并不是一个理想的方式，所以，设计器还提供了npm工具类，以供引入npm包。

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，展开资源管理页面。

步骤5 单击，进入添加工具类页面。

步骤6 工具类型选择“npm”。

步骤7 参考[表2-1](#)配置工具类的参数。

表 2-1 参数说明

参数	说明
名称	工具类名称。
包名	npm包名。
导出名称	import时的命名，如果是非解构，则可以自由命名，如果是解构，则npm包的导出必须要有该名称。
是否解构	解构则使用如import { export1 } from 'module'的方式导入。
入口路径	有些npm包的方法并不在默认导出中，如import { foo , bar } from "module-name/path/to/specific/un-exported/file"; 则需要填写入口路径。

参数	说明
版本号	npm包的版本号，需要符合npm包版本规范，详见 semver ，如不填则默认为latest。
CDN	如果页面使用了npm工具类，则需要手动录入npm cdn链接，否则可能会造成页面预览失败。

图 2-77 添加 function 工具类



步骤8 单击“保存”，完成function工具类添加。

----结束

cdn 链接相关说明

cdn链接就是npm包在浏览器直接可用的链接，如：<https://unpkg.inhuawei.com/>，提供了大部分的cdn链接，可以从上面获取。在页面预览中，并不会对所有代码和依赖进行转译，npm依赖需要以cdn方式进行引入，所以，如果添加了npm工具类，在预览失败时，请确保已添加cdn链接。

预置 cdn 链接说明


为了使用的方便，目前在Vue技术栈中，已经预置了一部分npm包的cdn链接，因此使用以下npm包时，无需添加cdn链接。

- @vueuse/core
- @vueuse/shared
- axios
- pinia
- vue
- vue-i18n
- vue-router
- vue/server-renderer
- @opentiny/vue

2.11.2 npm utils 使用示例

本章节以按钮组件单击时显示Loading效果为例为您介绍npm utils的使用。

添加组件

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
 - 步骤2** 在左侧导航栏中，选择“前端应用”。
 - 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
 - 步骤4** 在左侧插件栏中，单击，展开物料资产包。
 - 步骤5** 在物料资产包中选择按钮组件，并拖拽至中心画布中。
- 结束

添加 npm utils



- 步骤1** 在左侧插件栏中，单击，展开资源管理页面。
- 步骤2** 单击，进入添加工具类页面。
- 步骤3** 工具类型选择“npm”。
- 步骤4** 参考[表2-2](#)配置工具类的参数。

表 2-2 参数说明

参数	示例
名称	Loading
包名	@opentiny/vue
导出名称	Loading
是否解构	开启
入口路径	本示例不填

参数	示例
版本号	0.1.20
CDN	本示例不填

步骤5 单击“保存”。

----结束

绑定事件

步骤1 在中心画布中选中按钮组件。

步骤2 在右侧属性设置面板选择“高级”。

步骤3 鼠标悬停在“绑定事件”上，将显示事件列表。

步骤4 在事件列表中，单击OnClick事件。

步骤5 在弹框中输入方法名称，例如handleOnClick。

图 2-78 事件绑定



步骤6 单击“确定”，弹出JS面板。

步骤7 在JS面板中编写代码。

代码示例：

```
function handleOnClick(event) {
  this.loadingInstance = this.utils.Loading.service({
    text: "加载中",
    target: document.getElementById("tiny-loading1"),
  });
  setTimeout(() => {
    this.loadingInstance.close();
  }, 3000);
}
```

步骤8 单击“保存”，绑定事件完成。

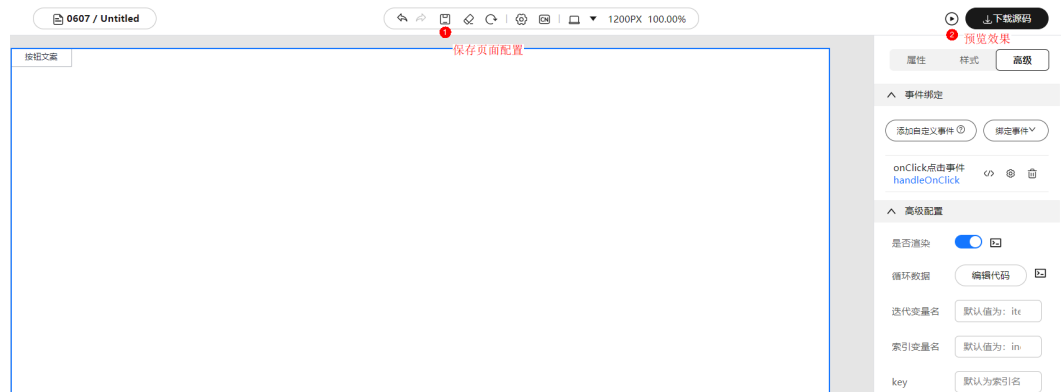
----结束

查看预览效果

步骤1 在顶部工具栏单击保存，保存页面配置。

步骤2 在顶部工具栏单击预览，进入预览页面。

图 2-79 预览页面



步骤3 单击预览页面的按钮，查看效果。


图 2-80 单击后效果



----结束

2.11.3 function utils 使用示例

添加 function utils

步骤1 在左侧插件栏中，单击 ，展开资源管理页面。

步骤2 单击 ，进入添加工具类页面。

步骤3 工具类型选择“function”。

步骤4 输入工具类的名称，并编写函数代码。

```
function isNil(value) {  
  return value == null  
}
```

图 2-81 添加 function 工具类



步骤5 单击“保存”，完成function工具类添加。

----结束

在 JS 面板的函数中使用 function utils

我们可以在JS面板或者页面生命周期函数中通过this.utils.isNil方式使用自定义的函数。

代码示例：

```
function xxxHandler(value) {  
  if(this.utils.isNil(value)) {  
    return  
  }  
  // ... other logic  
}
```

2.12 国际化资源管理

使用说明

同一个项目，可能需要同时支持多个语言，设计器提供了中英文切换，在能够一键切换语言，提升开发效率与开发体验。国际化是应用级别的，在任何一个页面都可以访问。

📖 说明

注意的是国际化只针对画布中的项目，不针对设计器本身。

添加国际化词条

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击🌐，展开国际化资源页面。

步骤5 单击“新增词条”。

步骤6 编辑key，中文和英文内容

📖 说明

- 创建时key会自动生成，也可以自定义输入。
- key需为唯一值，不可与现有key重复。
- 创建后key值不可修改。

图 2-82 新增词条



步骤7 单击页面空白处，完成国际化词条创建。

----结束

编辑国际化词条

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

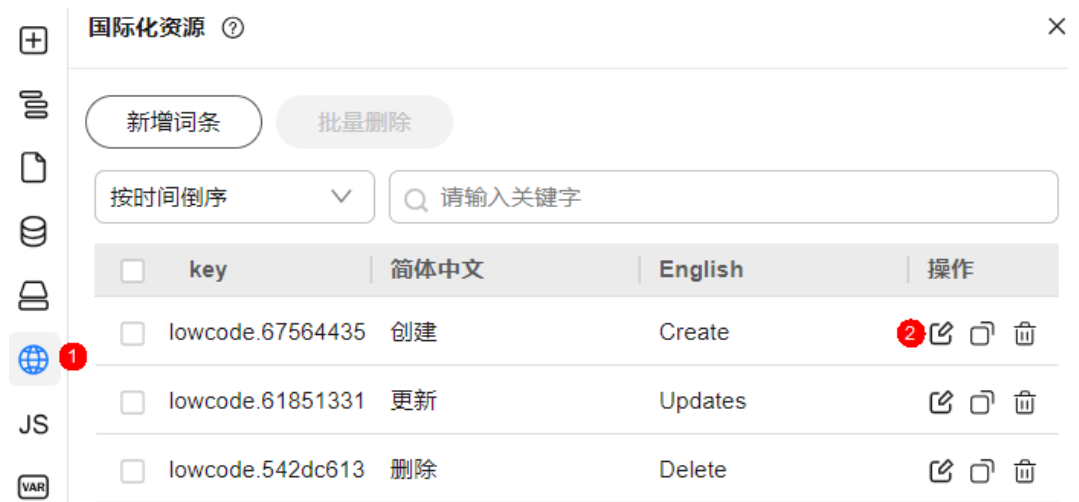
步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击🌐，展开国际化资源页面。

步骤5 选择待操作的词条，单击“操作”列的📄。

图 2-83 编辑词条



步骤6 重新编辑key，中文和英文内容。

步骤7 单击页面空白处，完成国际化词条编辑。


----结束


单个删除国际化词条

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

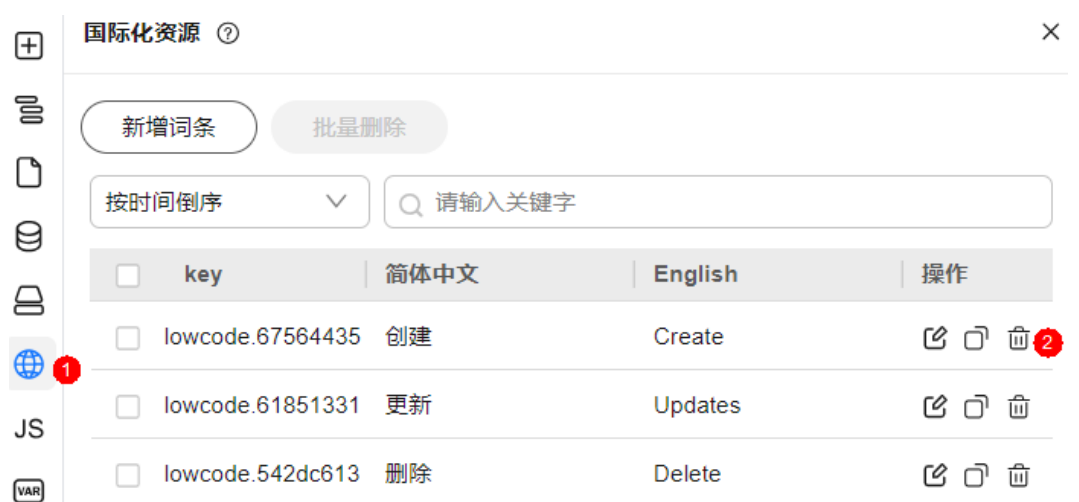
步骤3 单击应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击 ，展开国际化资源页面。

步骤5 选择待操作的词条，单击“操作”列的 。

步骤6 在弹框中单击“确定”，完成国际化词条删除。

图 2-84 删除词条



----结束

批量删除国际化词条


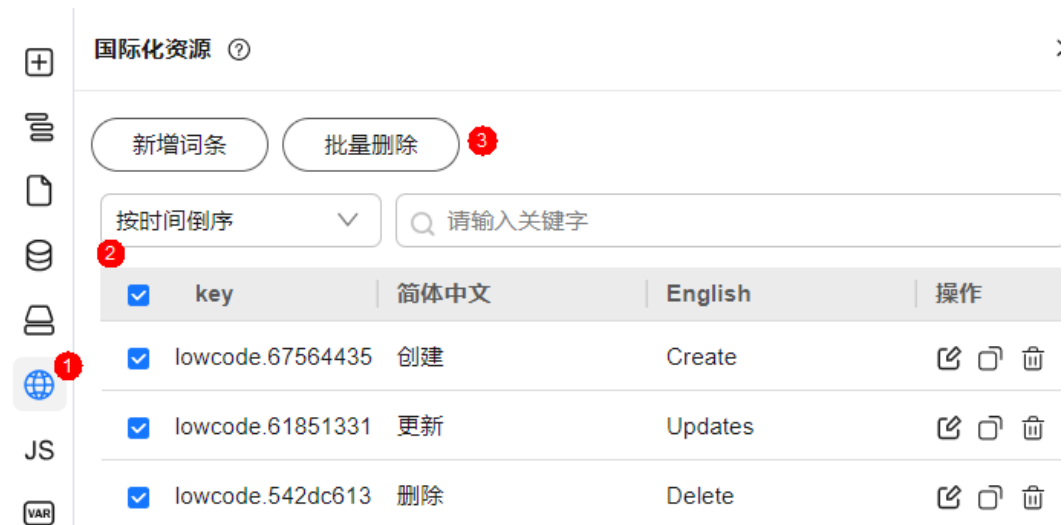
- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击，展开国际化资源页面。
- 步骤5 勾选待操作的词条，单击“批量删除”。

图 2-85 批量删除词条



- 步骤6 在弹框中单击“确定”，完成国际化词条批量删除。

----结束

复制词条键值



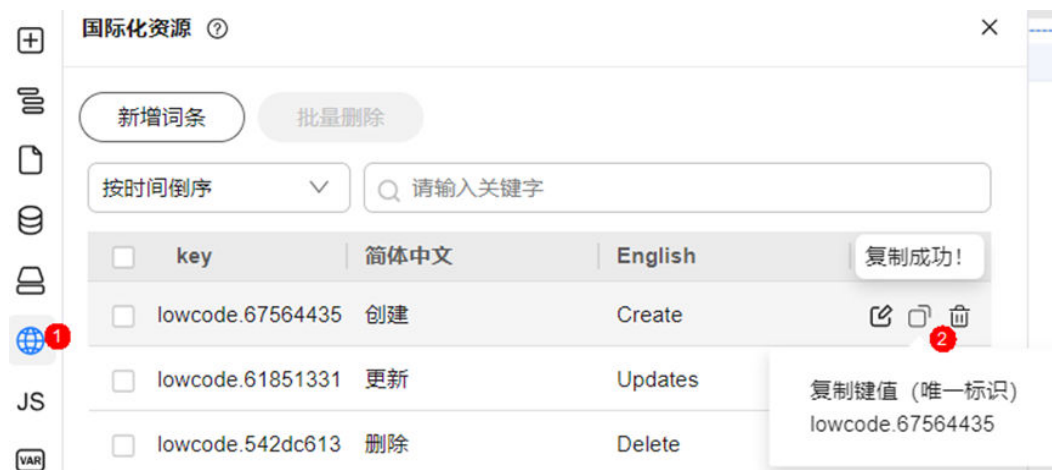
- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击，展开国际化资源页面。
- 步骤5 选择待操作的词条，单击“操作”列的。
页面显示“复制成功”，完成词条键值复制。

图 2-86 复制词条键值



----结束

使用国际化词条



- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击应用模块内的“应用开发”，进入设计器。
- 步骤4 在左侧插件栏中，单击 ，展开物料资产包。
- 步骤5 在物料资产包中选择组件，例如button组件，并拖拽至中心画布中。
- 步骤6 选中组件，在组件属性设置面板选择“属性”。
- 步骤7 选中文案参数旁的 ，进行变量绑定。
- 步骤8 输入变量t('lowcode.67564435')，其中lowcode.67564435为词条的key值，可参考[复制词条键值](#)获取。

图 2-87 绑定变量



- 步骤9 单击“确定”，变量绑定完成。
- 步骤10 单击顶部工具栏的中英文切换按钮，可进行中英文切换。

图 2-88 中文显示

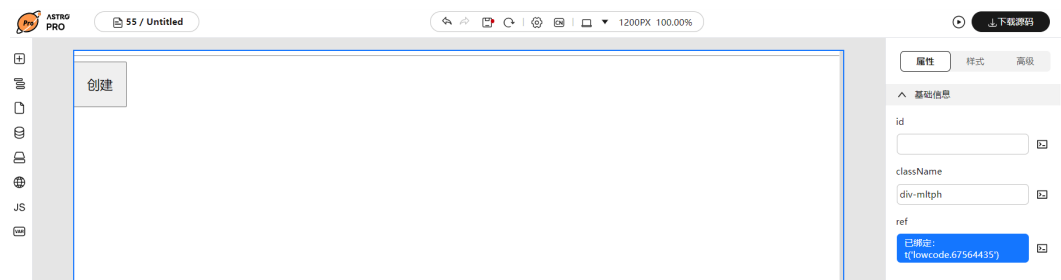
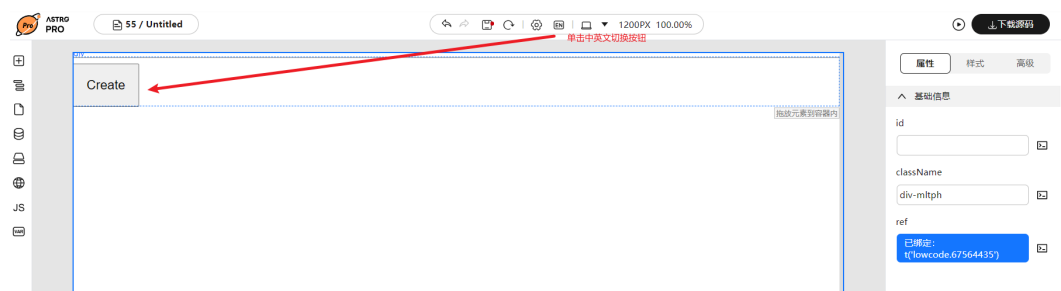


图 2-89 切换英语



----结束

2.13 使用 JS 面板

在常规代码开发中，我们需要为某个区块或者某个元素添加一些事件，比如单击事件，同一个页面的事件会统一保存到对应的页面JS中。

很多时候我们需要自定义一些方法去复用一些逻辑，也需要用到页面JS。

需要注意的是当前的页面JS只能使用声明函数，不能使用函数表达式声明函数，也不能在页面JS中定义其他的变量。

页面JS中可以通过this访问当前页面的state，全局的stores，以及t函数（获取国际化词条）。

下面将通过两个示例分别展示如何使用。

为组件添加单击事件

- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“前端应用”。
- 步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4 选中组件，在组件属性设置面板选择“高级”。
- 步骤5 鼠标悬停在“绑定事件”上，将显示事件列表。
- 步骤6 在事件列表中，选择onClick事件。
- 步骤7 在弹框中设置绑定事件。
 - 方法名称：选择已有方法或者添加新方法，如果选项添加新方法，单击“确定”之后将在JS面板中创建一个该名称的新方法。

- 扩展参数：调用当前事件传入的真实参数，数组格式，追加在原有事件参数之后如：onClickNew(eventArgs, extParam1, extParam2, ...)。

图 2-90 设置绑定事件



步骤8 单击“确定”，弹出JS面板，刚创建的onClickNew事件已存在JS面板中。

步骤9 在JS面板中编写绑定方法的具体逻辑。

图 2-91 实现逻辑编写




步骤10 单击“保存”，绑定事件完成。


----结束

创建普通方法在生命周期中使用

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 单击待编辑应用模块内的“应用开发”，进入设计器。
- 步骤4** 在左侧插件栏中，单击**JS**，展开JS页面。

步骤5 在JS页面中添加getA方法。

步骤6 在左侧插件栏中，单击 ，进入“页面管理”页面。

步骤7 鼠标悬浮到当前页面名称上，将显示操作按钮，单击 ，进入页面设置页面。

步骤8 单击“添加页面生命周期”，选择setup。

步骤9 在setup函数使用getA()，单击“保存”。

----结束

出码结果

在顶部工具栏单击“下载源码”，查看出码结果。

```
<template>
  <div>
    <span>{{ state.testValue }}</span>
     getA(eventArgs, 12)"
    />
  </div>
</template>

<script setup>
import * as vue from 'vue'
import { defineProps, defineEmits } from 'vue'
import { I18nInjectionKey } from 'vue-i18n'

const props = defineProps({})
const emit = defineEmits({})

const { t, lowcodeWrap, stores } = vue.inject(I18nInjectionKey).lowcode()
const wrap = lowcodeWrap(props, { emit }, t)

const state = vue.reactive({
  testValue: {}
})

const getA = wrap(function (eventArgs, args0) {
  const testData = {
    name: 'rico',
    age: 18
  }
  return testData
})

wrap({
  stores,
  state,
  getA
})

const setup = wrap(function setup({ props, state, watch, onMounted }) {
  state.testValue = getA()
})
setup({ props, context: { emit }, state, …vue })
</script>
```

如上代码所示：在设计器中编辑的方法，为组件添加的单击事件等都有生成，并且会默认暴露出t函数（国际化，具体可参考[如何使用国际化词条](#)）和全局stores(pinia)状态。

2.14 变量管理

2.14.1 添加页面变量

使用说明


页面状态变量仅适用于当前页面。

操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“前端应用”。

步骤3 单击待编辑应用模块内的“应用开发”，进入设计器。

步骤4 在左侧插件栏中，单击，展开状态管理页面。

步骤5 选择“页面状态”，单击“添加变量”。

步骤6 设置变量基本信息。

- 变量名：输入变量名称。
- 初始值类型：可选择“JSON类型”和“JS表达式类型”
- 初始值：数据写法和JS写法一致。
 - 字符串: "string"
 - 数字: 123
 - 布尔值: true/false
 - 对象: {"name": "xxx"}
 - 数组: ["1", "2"]
 - 空值: null
 - JS表达式: (需要先选择JS表达式类型)
 - 示例1: `t('i18nkey1')`
 - 示例2: `function fnName() {}`
 - 示例3: `{ getValue: () => {} }`

注意：使用JS表达式定义state变量的时候无法调用state其他变量定义，另由于JS函数定义在变量之后，也无法调用JS面板定义的函数。

图 2-92 添加变量



步骤7 (可选) 设置变量高级配置。

- **getter**: 用于获取 (读取) 类的私有属性的值。Getter方法通常没有参数, 并且返回属性的值。

示例:

```
function getter() {
  // this.state.name = `${this.props.firstName} ${this.props.lastName}`
}
```

- **set**: 用于设置 (写入) 类的私有属性的值。Setter方法通常接受一个参数, 该参数是要设置的新值, 并且可能包括一些逻辑来验证或处理这个值, 然后才将其赋给属性。示例:

```
function setter() {
  // const [firstName, lastName] = this.state.name.split(' ')
  // this.emit('update:firstName', firstName)
  // this.emit('update:lastName', lastName)
}
```

步骤8 单击“保存”, 完成变量添加。


----结束

2.14.2 添加全局变量

使用说明

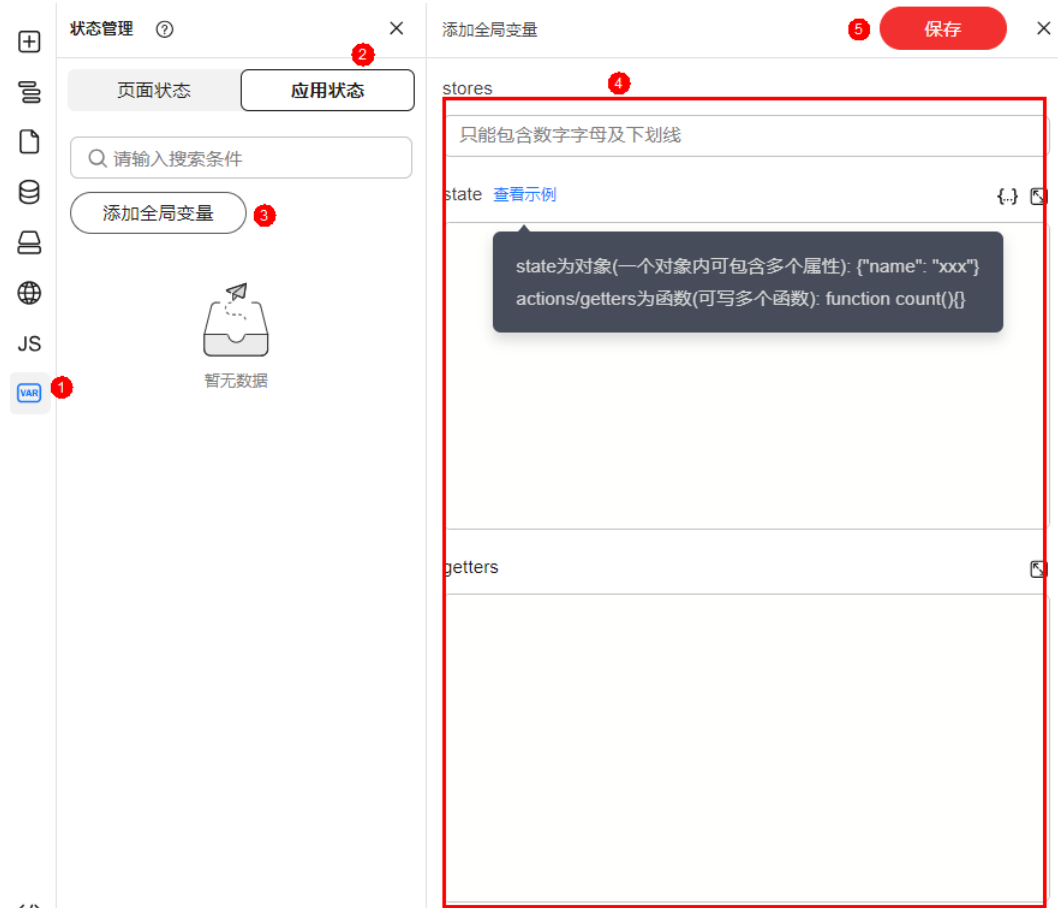
全局变量是在整个应用中都可访问的变量。它们的作用域是全局的, 可以在应用的任何一个页面调用。

操作步骤

- 步骤1 参考[登录AstroPro界面](#)中操作, 登录AstroPro界面。
- 步骤2 在左侧导航栏中, 选择“前端应用”。
- 步骤3 单击待编辑应用模块内的“应用开发”, 进入设计器。
- 步骤4 在左侧插件栏中, 单击, 展开状态管理页面。
- 步骤5 选择“应用状态”, 单击“添加全局变量”。
- 步骤6 设置变量基本信息。

- stores: 是包含所有状态 (state)、视图 (view) 和行为 (actions) 的容器。store属性名称只能以字母或下划线开头, 且仅包含数字、字母及下划线。
- state: 是store中存储的应用程序状态, 通常是响应式的数据对象。
- actions: 是store中的计算属性, 允许开发者从state派生出一些状态。
- getters: 是store中的方法, 用于提交mutations或执行异步操作。

图 2-93 添加全局变量



步骤7 单击“保存”，完成变量添加。

----结束

2.15 生成业务代码

使用说明

页面设计完成后, 您可以根据配置生成应用的基本代码。代码生成后, 会下载至本地, 供您使用。

操作步骤

步骤1 参考[登录AstroPro界面](#)中操作, 登录AstroPro界面。

- 步骤2** 在左侧导航栏中，选择“前端应用”。
- 步骤3** 选择待操作应用，单击模块内的“应用开发”，进入设计器。
- 步骤4** 单击顶部工具栏的“下载源码”按钮。
- 步骤5** 选择下载路径。
- 步骤6** 选择生成到本地的文件。

您可以选择全量或指定部分文件内容。

图 2-94 选择生成文件



- 步骤7** 单击“确定”，代码将下载至本地路径。

----结束

3 后台应用管理

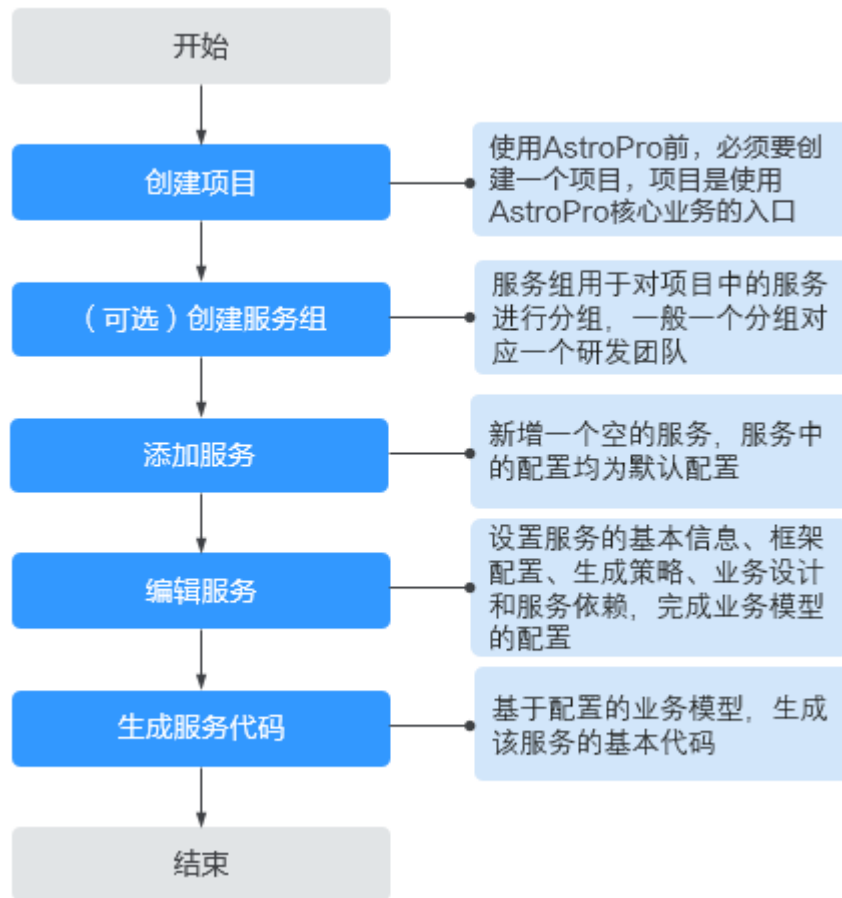
3.1 创建企业核心应用

3.1.1 了解构建流程

在AstroPro中，用户通过业务建模，可生成高可用、高可靠、以及安全稳定的企业级IT应用框架。业务建模是指通过业务设计，将实际业务涉及的对象和行为转换为元数据中的对象、对象关系、服务依赖等构成的模型，通过模型生成服务，实现业务需求。

使用AstroPro创建企业核心应用的流程，如[图3-1](#)所示。

图 3-1 创建企业核心应用流程图



- 1. 创建项目**
项目是使用AstroPro核心业务的入口。在使用AstroPro前，需要先创建一个项目。
- 2. 创建服务组**
服务组用于对项目中的服务进行分组，一般一个分组对应一个研发团队。创建项目后，默认会创建一个和项目同名的服务组，所有新建服务默认在此分组下。
- 3. 添加服务**
在新增服务界面，通过简单的配置，完成服务框架的搭建。
- 4. 编辑服务**
添加服务的操作，相当于为服务搭建了一个框架。如果需要服务实现某些特定的功能，还需要您根据业务需求，对服务进行业务模型配置。
- 5. 生成服务代码**
基于配置的业务模型，生成服务的基本代码。代码生成后，会提供一个压缩包，供您直接使用。

3.1.2 步骤 1：创建项目

项目是一个功能相对完备的业务系统，通常情况下由一个或多个服务组组成。项目是使用AstroPro核心业务的入口。在AstroPro中会为每个租户提供一个工作空间，您可以在工作空间中新建项目。项目创建后，您可以为其他用户添加项目的访问权限，详情请参见[角色管理](#)。

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 项目”，单击“新建项目”。

首次进入项目时，请按照界面提示，开通工作空间。开通工作空间后，即可在该工作空间内创建项目。

步骤3 设置项目的基本信息，单击“确定”。

图 3-2 设置项目的基本信息

基本信息

* 项目名称

AstroProject

自动创建同名服务组

描述

请输入项目描述

版权信息

* Package

com.astropro

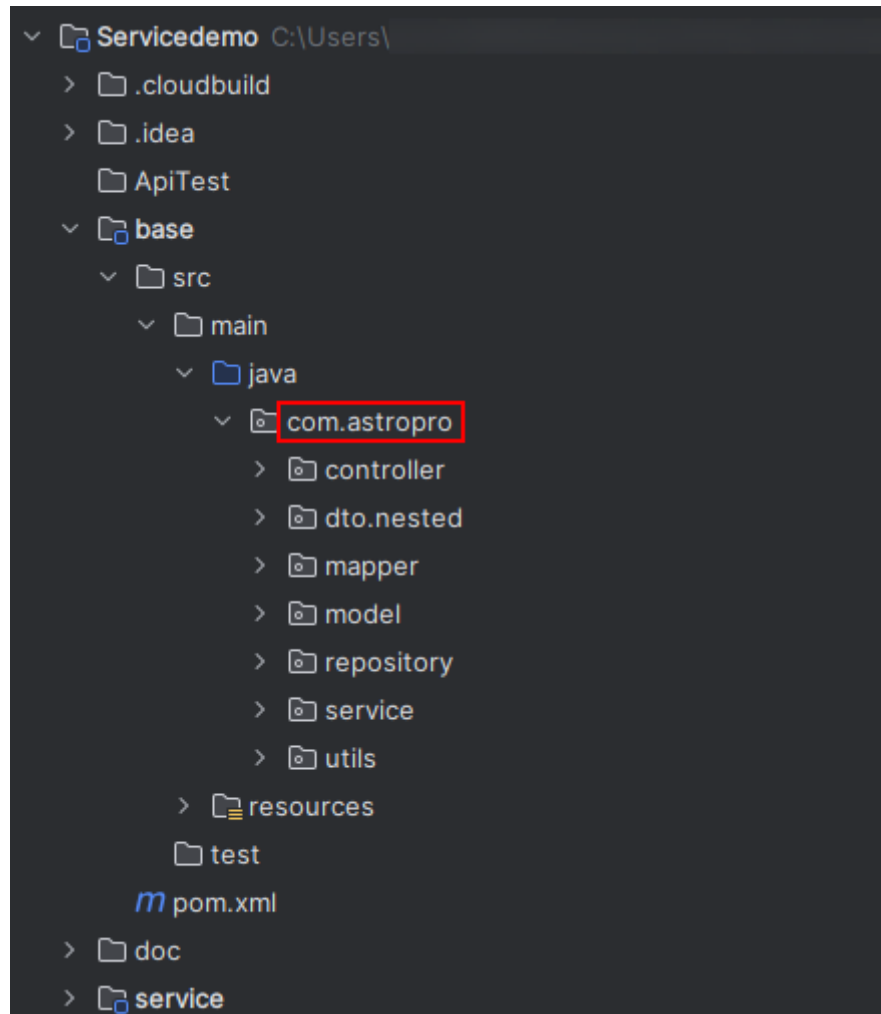
* Group

grouppro

- 项目名称：新增项目的名称，只能包含大小写字母、数字、连字符（-）和下划线（_）。
- 描述：设置项目的描述信息，通常设置为项目的用途或者功能。
- 版权信息：自定义代码的版权信息，请以“//”开头或者以“/*”开头并以“*/”结尾。

- Package: 设置生成代码的顶层包名，由一个或多个小写字母和数字组成，片段之间用点号 (.) 连接，且必须以小写字母开头。

图 3-3 顶层包名



- Group: 设置项目的默认组名，只能包含大小写字母、数字、连字符 (-)、下划线 (_) 和点 (.)。

----结束

3.1.3 (可选) 步骤 2: 创建服务组

服务组用于对项目中的服务进行分组，一般一个分组对应一个研发团队。创建项目后，默认会创建一个和项目同名的服务组，所有新建服务默认在此分组下。您也可以不使用默认的服务组，直接新建一个服务组。

步骤1 参考**步骤1: 创建项目**中操作，创建一个项目。

步骤2 项目创建后，在左侧导航栏中，选择“项目管理 > 服务组”。

步骤3 在项目后的下拉框中，选择**步骤1**中创建的项目，单击“新建服务组”。

步骤4 设置服务组的基本信息，单击“确定”。

图 3-4 设置服务组基本信息

< 新建服务组

基本信息

* 服务组名

 ❗

描述

取消 确定 ❗

- 服务组名称：设置新建服务组的名称，只能包含大小写字母、数字、连字符（-）和下划线（_）。
- 描述：输入服务组的描述信息，通常设置为服务组的用途或者功能。

----结束

3.1.4 步骤 3：添加服务

在AstroPro中，快速添加一个服务，新增服务中的配置均采用默认配置。此处新建服务的操作，相当于为服务搭建了一个框架。

步骤1 在AstroPro界面的左侧导航栏中，选择“项目管理 > 服务”。

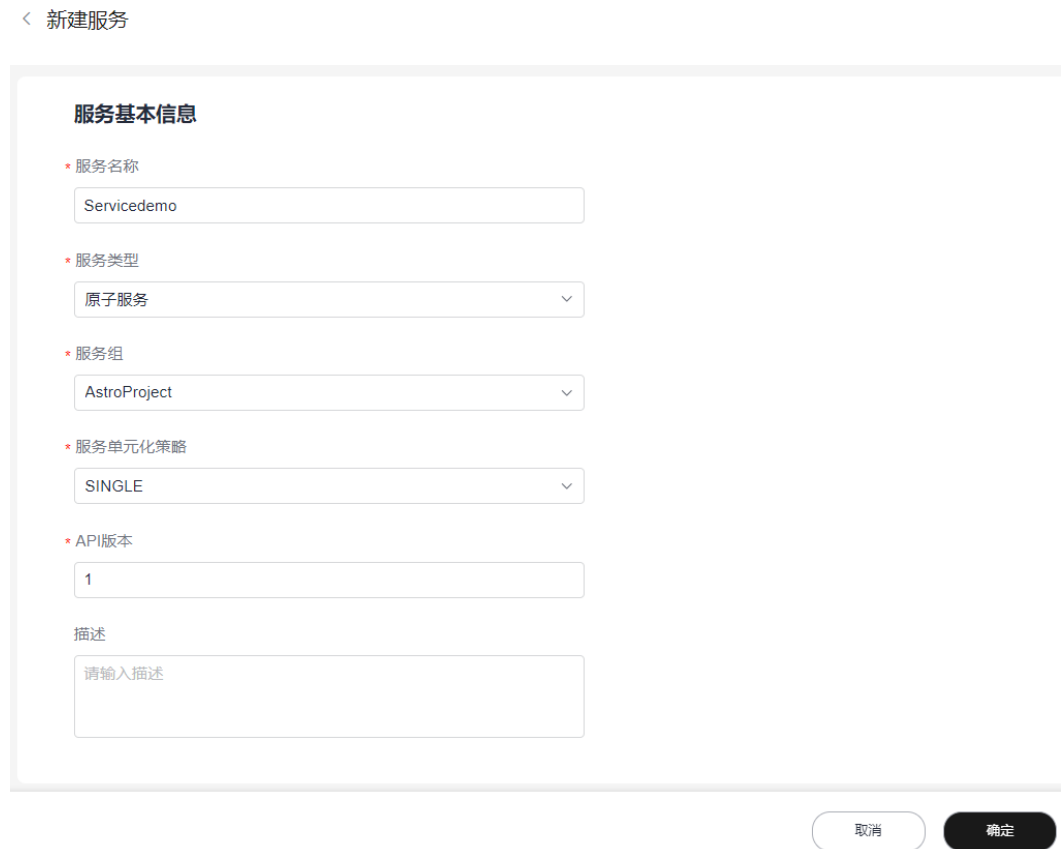
步骤2 选择已创建的项目和服务组，单击“新建服务”。

图 3-5 新建服务



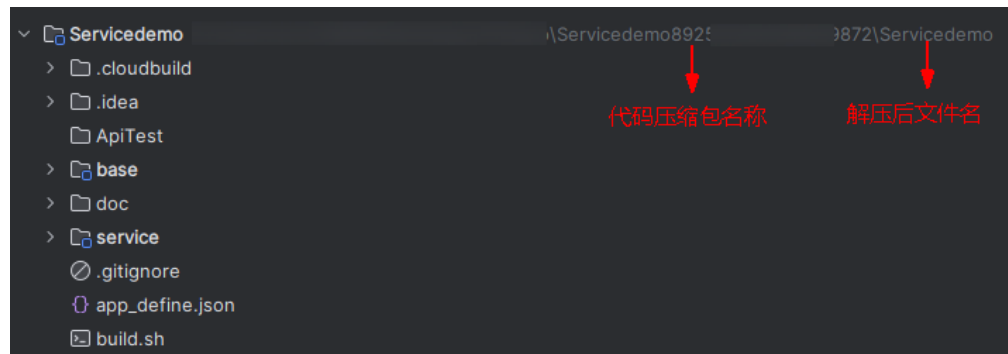
步骤3 设置服务的基本信息，单击“确定”。

图 3-6 设置服务基本信息



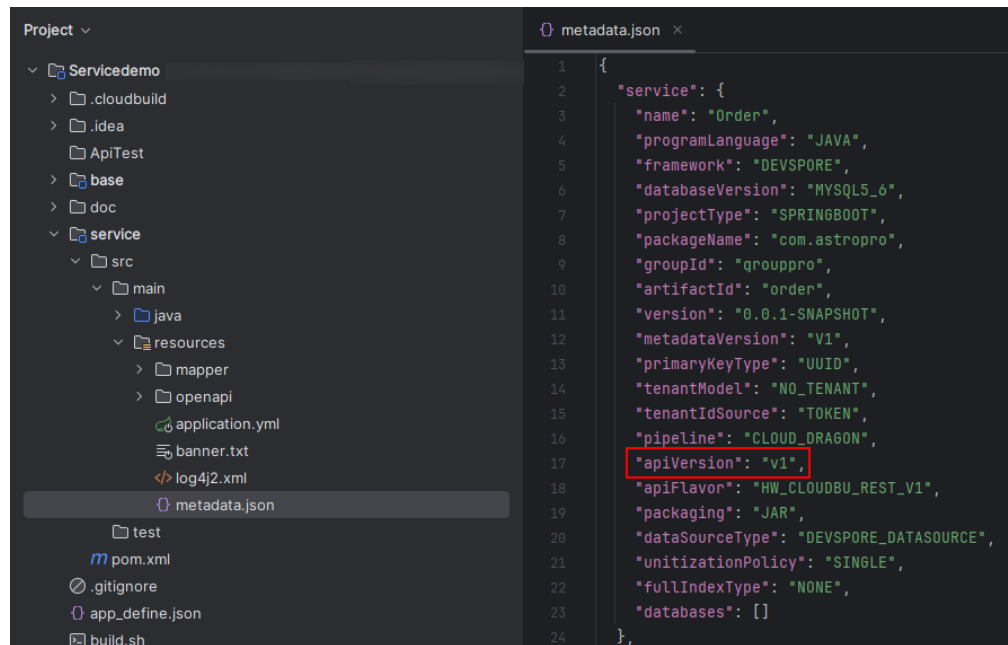
- 服务名称：设置待添加服务的名称，生成的服务代码压缩包和解压后的文件会以此命名。服务名称由英文字母、数字或“-”组成，且必须以字母开头，一般采用驼峰格式，长度最低为两位。

图 3-7 代码压缩包名称



- 服务类型：当前仅支持创建原子服务。原子服务是指对外提供业务对象管理API，有独立数据存储（一般为独立数据库）的服务。原子服务之间可以相互调用。
- 服务组：选择服务所属的分组，即（可选）步骤2：创建服务组中创建的服务组。
- 服务单元化策略：服务在子域内的单元化策略。服务单元化策略必须在一个子域内定义，不能跨子域。当应用比较复杂时，可基于领域的特定概念将应用分解为多个领域，每个领域就是一个子域，如核心子域、支撑子域和通用子域。
创建服务时，仅支持SINGLE，即单库，无论子域是否进行单元化部署，该服务只在一个单元（一般以region为单元）内部署。服务创建后，在步骤4：编辑服务中编辑元数据时，可进行修改。
- API版本：指定服务的API版本，对应服务metadata.json文件中，Service段的apiVersion字段，一般为v1、v2类型的值。

图 3-8 apiVersion



- 描述：设置服务的描述信息。

----结束

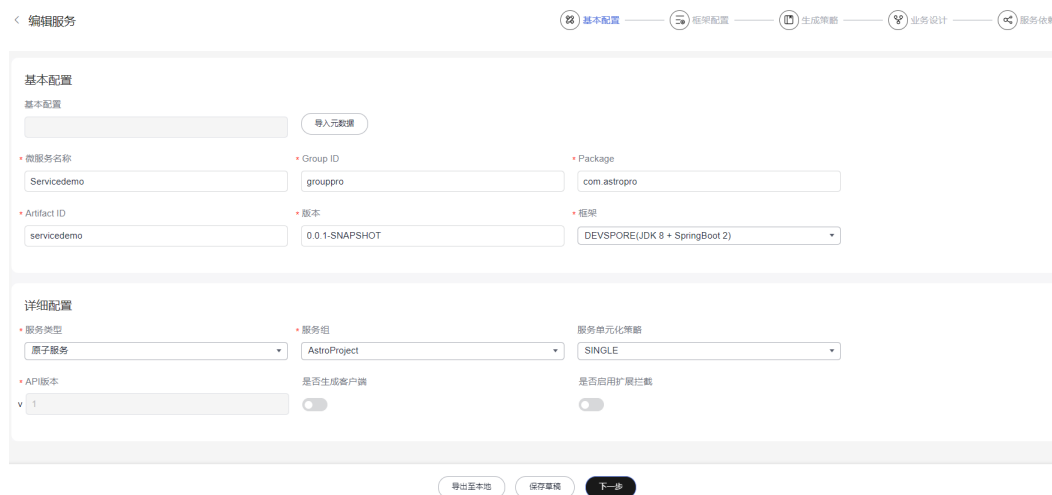
3.1.5 步骤 4：编辑服务

在AstroPro中，用户通过业务建模，可生成高可用、高可靠及安全稳定的企业级IT应用框架。**步骤3：添加服务**中的操作，相当于为服务搭建了一个框架，如果需要通过某些特定的功能，还需要您根据自身业务需求进行业务模型配置。

步骤1 在服务列表中，单击**步骤3：添加服务**中已创建服务后的“编辑”。

步骤2 参考**编辑服务**中操作，完成服务的配置。

图 3-9 服务配置



说明

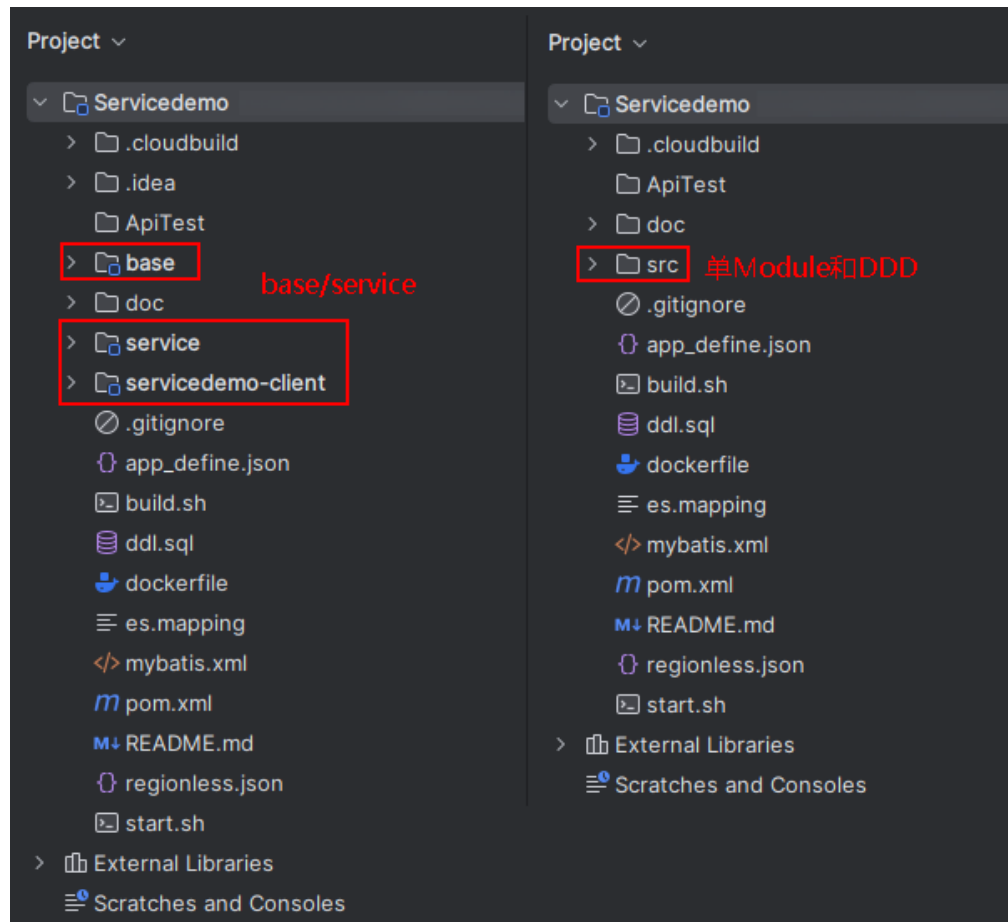
购买AstroPro专业版实例时，才会显示“是否生成客户端”和“是否启用扩展拦截”这两个配置项。

- 基本配置、框架配置和生成策略：请根据自身业务需求直接在界面勾选，不同的配置会呈现不同的效果。例如，“生成策略 > 代码风格 > 工程目录”设置不同，生成的代码目录结构也会有所不同。

图 3-10 设置代码工程目录结构



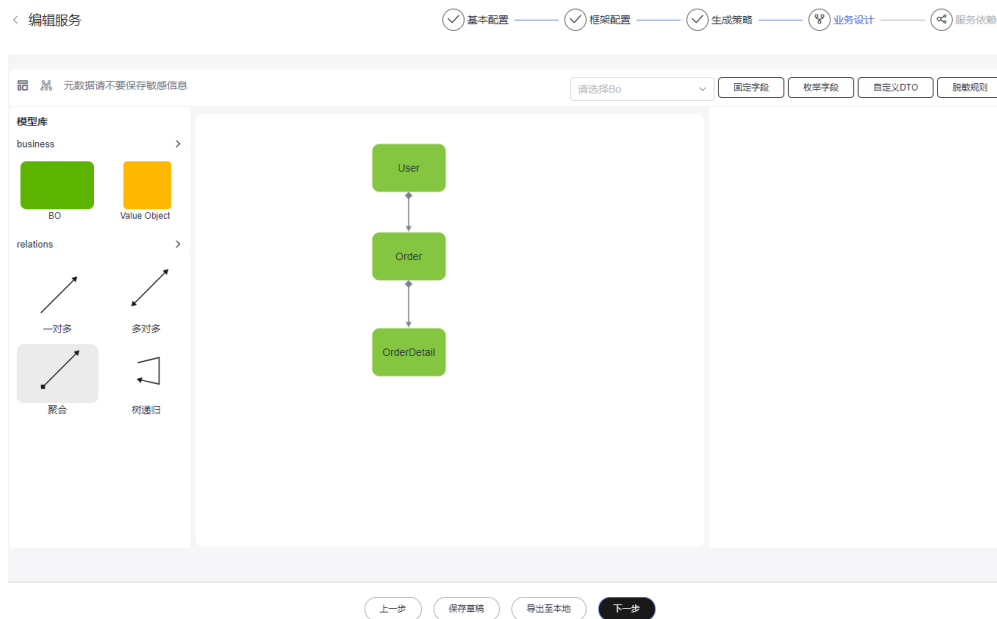
图 3-11 单 Module 和 base/service 生成代码目录效果



- 业务设计：AstroPro提供的核心能力，是用户设计业务的基础。通过添加对象、设置对象属性和为对象建立对应的关系，来实现某些特定的功能。业务设计过程中，使用到的对象及对象间关系介绍，请参见[对象详解](#)。

例如，某个订单系统中包括用户（User）、订单（Order）和订单详情（OrderDetail）三个业务对象，且三个对象之间存在聚合关系，即用户存在时，订单才会存在，订单存在时，订单详情才会存在。

图 3-12 订单业务设计



- 服务依赖：通常情况下，一个应用不是一个单独的服务，可能由多个服务共同组成。这些服务之间可能存在一些跨服务的调用，此时就需要通过添加依赖服务，把这些服务的客户端集成过来。

说明

请确保被依赖的服务已开启“是否生成客户端”配置，否则添加依赖服务时会报错。

图 3-13 添加服务依赖



图 3-14 开启“是否生成客户端”配置



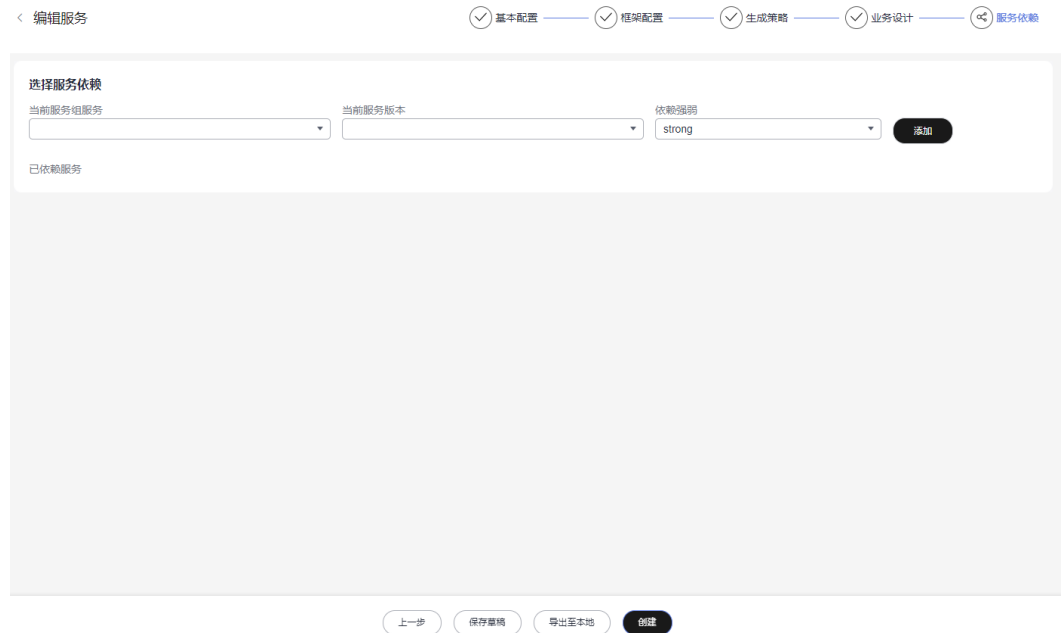
----结束

3.1.6 步骤 5：生成服务代码

根据配置的业务模型生成该服务的基本代码。代码生成后，会提供一个压缩包，供您使用。关于压缩包中代码的详细介绍，请参见[服务开发框架详解](#)。

步骤1 在服务依赖页面，单击“创建”。

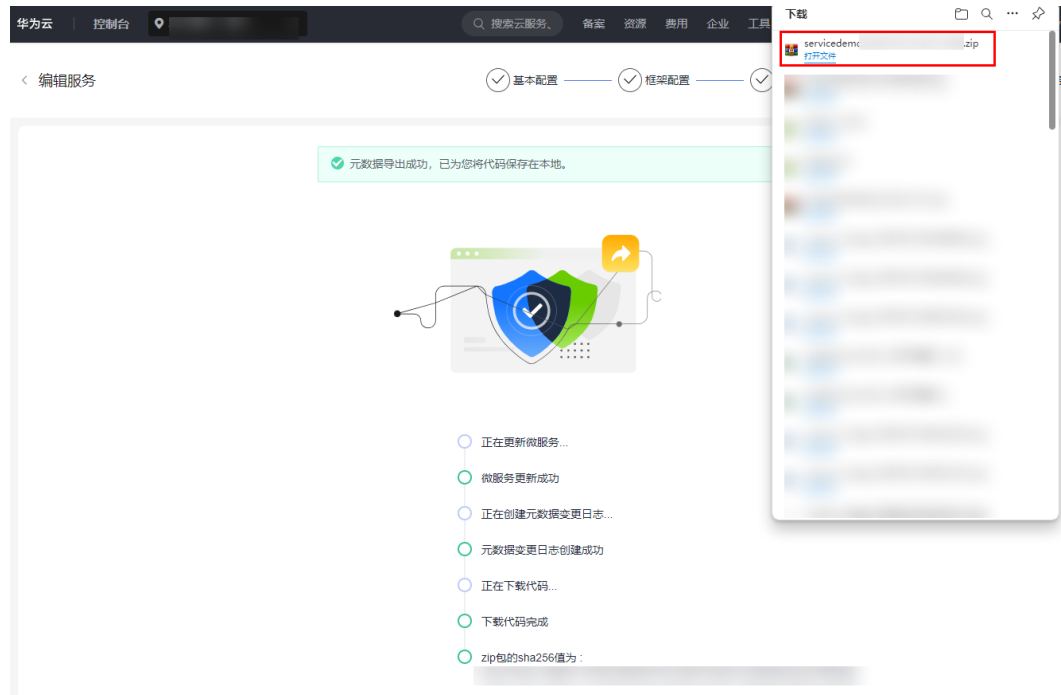
图 3-15 创建服务



步骤2 输入变更日志描述信息，单击“创建”。

系统开始创建服务，并生成该服务的基本代码。代码生成后，界面会提供一个压缩包，可直接下载使用。压缩包格式为“服务名称+唯一ID”。

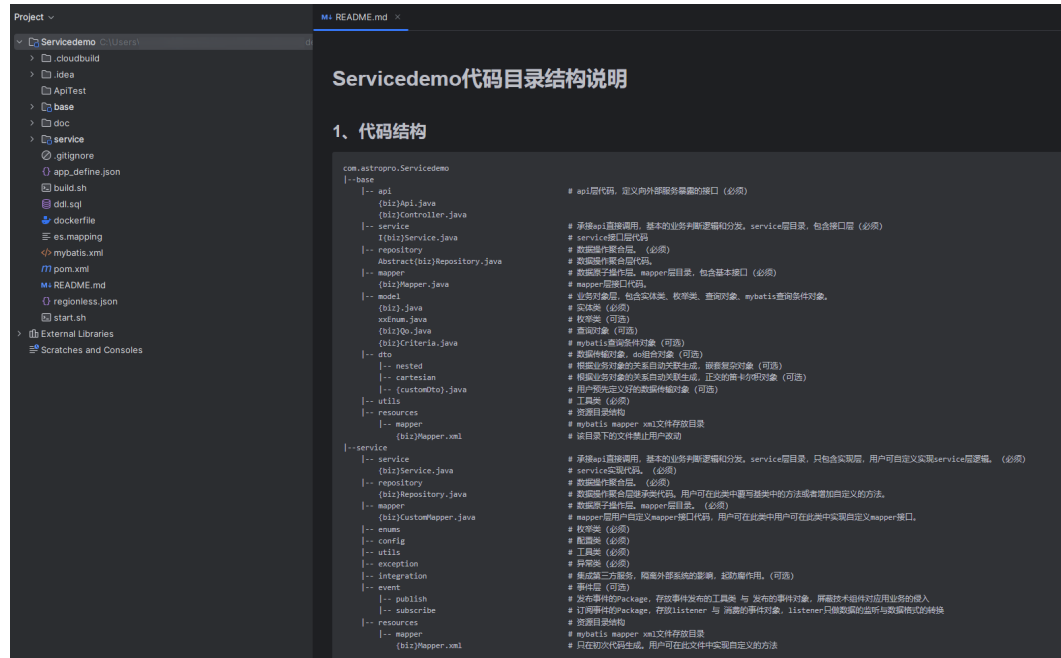
图 3-16 开始创建微服务



步骤3 到此您已完成整个企业核心应用的创建。

打开生成的服务代码包，在“README.md”文件中可查看代码目录结构的说明，如图3-17所示。

图 3-17 代码目录结构说明



代码目录结构分为“base/service”、“单Module”和“DDD”三种，在编辑元数据的“生成策略 > 代码风格 > 工程目录”中可进行定义，生成效果差异如图3-19所示。本示例采用默认配置即“base/service”样式，关于“单Module”和“DDD”样式的代码目录结构说明，请参见[服务开发框架详解](#)。

图 3-18 设置代码风格

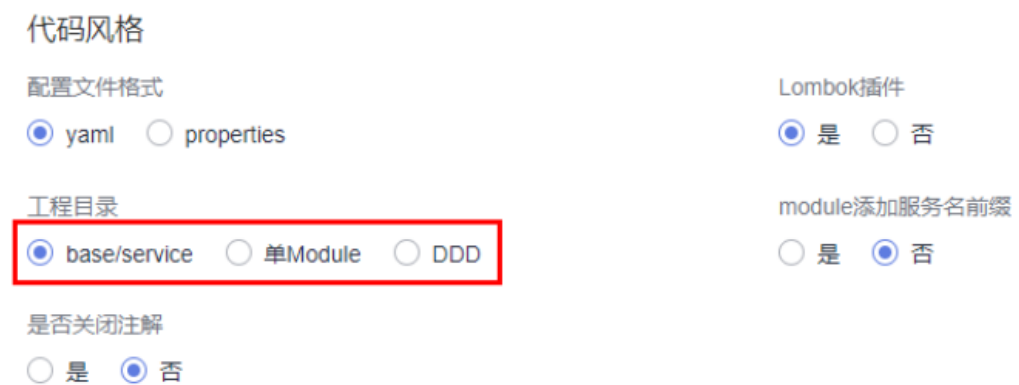
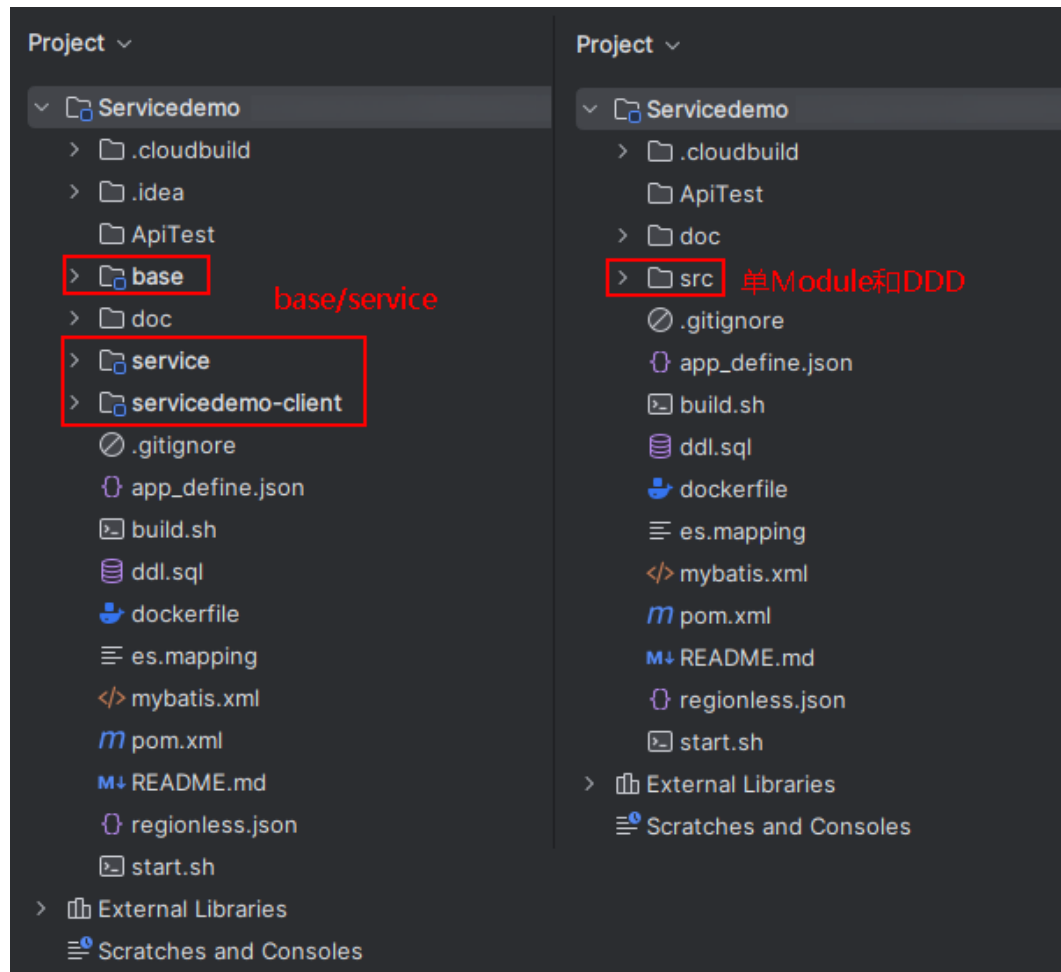


图 3-19 工程目录不同类型设置效果



----结束

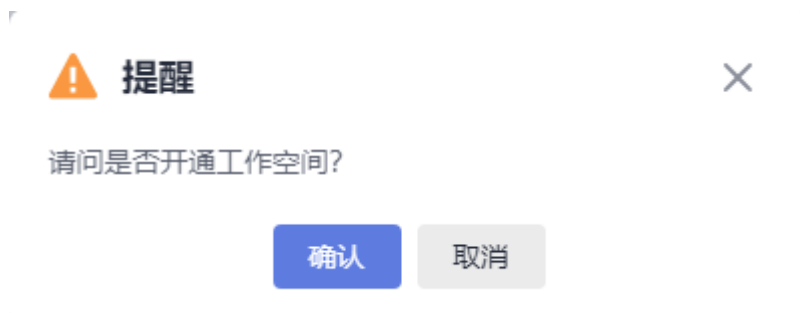
3.2 项目管理

3.2.1 新建项目

使用说明

项目是一个功能相对完备的业务系统，通常情况下由一个或多个服务组组成。项目是使用AstroPro核心业务的入口。在AstroPro中会为每个租户提供一个工作空间，您可以在工作空间中新建项目。首次进入项目时，请先开通工作空间。

图 3-20 开通工作空间



操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 项目”，单击“新建项目”。
- 步骤3** 设置项目的基本信息，单击“确定”。

图 3-21 设置项目的基本信息

基本信息

* 项目名称

AstroProject

自动创建同名服务组

描述

请输入项目描述

版权信息

* Package

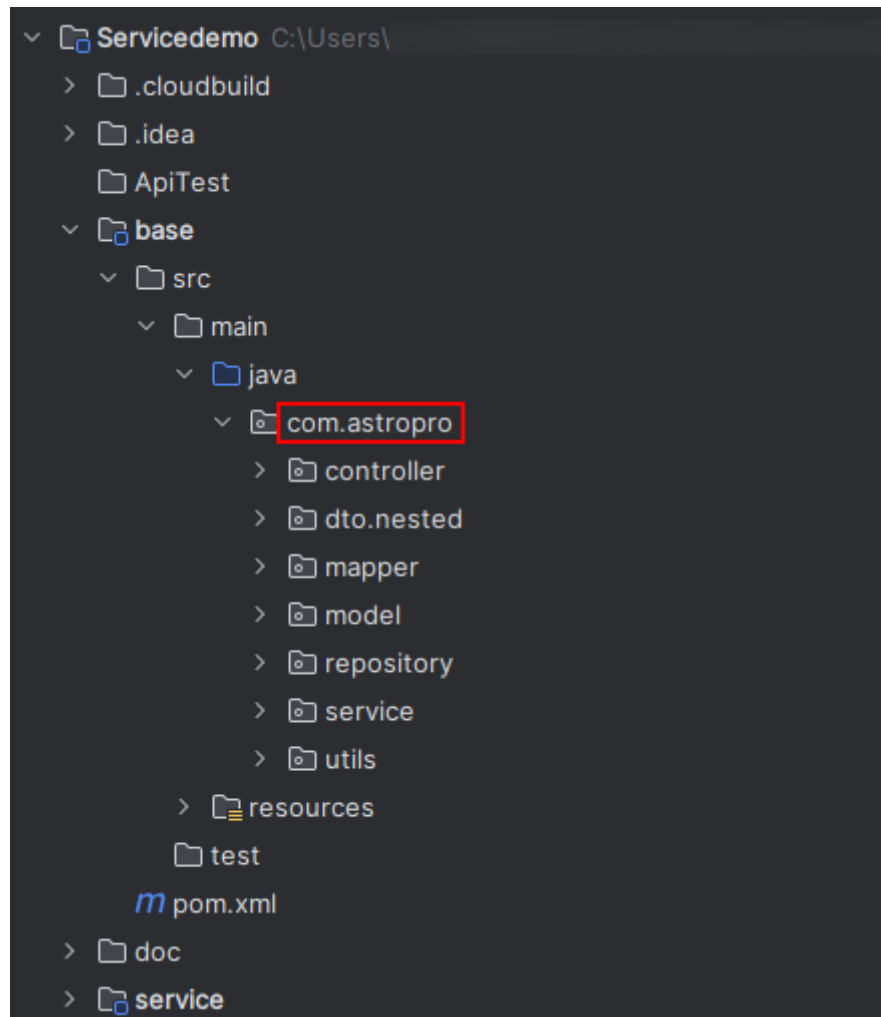
com.astropro

* Group

grouppro

- 项目名称：新增项目的名称，只能包含大小写字母、数字、连字符（-）和下划线（_）。
- 描述：设置项目的描述信息，通常设置为项目的用途或者功能。
- 版权信息：自定义代码的版权信息，请以“//”开头或者以“/*”开头并以“*/”结尾。
- Package：设置生成代码的顶层包名，由一个或多个小写字母和数字组成，片段之间用点号（.）连接，且必须以小写字母开头。

图 3-22 顶层包名



- Group: 设置项目的默认组名，只能包含大小写字母、数字、连字符 (-)、下划线 (_) 和点 (.)。

----结束

3.2.2 编辑项目

使用说明

项目创建后，支持再次编辑项目的基本信息。创建项目时会自动创建一个和项目同名的服务组，修改项目名称时，已创建的服务组名称不会随之改动。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 项目”。
- 步骤3** 在项目列表中，单击[新建项目](#)中已创建项目后的“编辑”。
- 步骤4** 按需修改项目的基本信息，单击“确定”。

图 3-23 修改项目的基本信息

< 编辑项目

基本信息

* 项目名称

描述

版权信息

* Package

* Group

- 项目名称：新增项目的名称，只能包含大小写字母、数字、连字符（-）和下划线（_）。
- 描述：设置项目的描述信息，通常设置为项目的用途或者功能。
- 版权信息：自定义代码的版权信息，请以“//”开头或者以“/*”开头并以“*/”结尾。
- Package：设置生成代码的顶层包名，由一个或多个小写字母和数字组成，片段之间用点号（.）连接，且必须以小写字母开头。
- Group：设置项目的默认组名，只能包含大小写字母、数字、连字符（-）、下划线（_）和点（.）。

----结束

3.2.3 删除项目

使用说明

删除项目前，请确保已删除项目中的服务组。如何删除服务组，请参见[删除服务组](#)。

单个删除项目

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 项目”。
- 步骤3** 在项目列表中，单击待删除项目后的“删除”。
- 步骤4** 在弹出的确认框中，单击“确认”，即可删除项目。
项目删除后不可恢复，请谨慎操作。

----结束

批量删除项目

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 项目”。
- 步骤3** 勾选待删除的项目，单击“批量删除”。
- 步骤4** 在弹出的确认框中，单击“确认”，即可批量删除项目。
项目删除后不可恢复，请谨慎操作。

----结束

3.2.4 角色管理

3.2.4.1 了解 AstroPro 中角色

AstroPro中的角色包括工作空间级角色和项目级角色两种，您可以通过为不同的用户赋予不同的角色，来控制用户对AstroPro的操作权限。一个用户在AstroPro中只能拥有一种角色。

工作空间管理员

账号登录AstroPro后会自动开通工作空间，每个账号只能开通一个工作空间，开通工作空间的账号默认为此空间的工作空间管理员。工作空间管理员对本工作空间下的所有资源具有增删改查权限，可以为本账号下用户分配项目下的角色。每个工作空间至少拥有一个工作空间管理员。

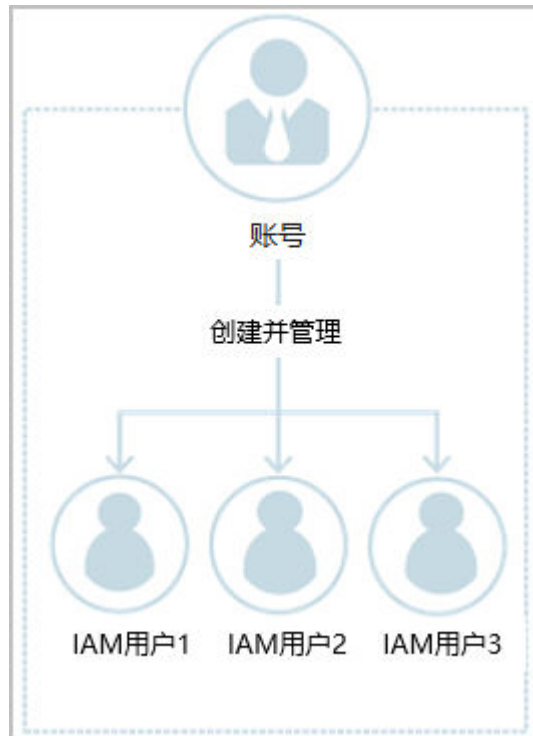
账号是指当您首次使用华为云时注册的账号，该账号是您的华为云资源归属、资源使用计费的主体，对其所拥有的资源及云服务具有完全的访问权限，可以重置用户密码、分配用户权限等。账号统一接收所有IAM用户进行资源操作时产生的费用账单。在一个账号下，您可以创建多个用户，账号和用户的关系如[图3-24](#)所示。如果您没有华为账号，可参考[注册华为账号并开通华为云](#)中操作注册。如何在账号中添加用户，请参见[创建IAM用户](#)。

说明

账号下的用户，如果没有分配任何AstroPro中的角色，则只能查看工作空间下的所有资源，不具备其他权限。

例如，给“IAM用户1”赋予工作空间管理员权限，给“IAM用户2”赋予项目A中的项目管理员权限，则“IAM用户1”可以增删改查本工作空间下的所有资源，“IAM用户2”仅可对工作空间中的项目A执行增删改查等操作，而“IAM用户3”未赋予任何角色，则只能查看本工作空间下的资源。

图 3-24 华为账号与 IAM 用户



项目级角色

除了工作空间级角色外，AstroPro还为每个项目预置了项目管理员、架构师和开发者三种类型的角色。

- 项目管理员
 - 可以增删改查项目下的所有资源。
 - 可以为项目下的所有角色分配用户（仅限于同账号下的用户，如图3-24中的IAM用户1、IAM用户2）。
- 架构师
 - 可以新建服务。
 - 可以删除自己创建的服务，不能删除其他人创建的服务。
 - 可以对项目下的所有服务进行编辑。
- 开发者
 - 可以查看项目下的所有资源。
 - 可以重新编译有限项目下的服务并下载代码。

AstroPro 角色说明

不同的角色，操作权限有所不同，AstroPro常用操作与角色之间的关系如表3-1所示，您可以按需申请您的角色权限。

说明

如果您申请的新权限低于您当前的权限级别，系统将只保留一个角色权限，即新申请的权限会覆盖现有的权限。因此，在您决定更换权限之前，建议您仔细了解并比较您当前的权限设置与新申请权限之间的具体差异。

表 3-1 AstroPro 常用操作与角色之间的关系

操作	工作空间管理员	项目管理员	架构师	开发者	其他（未授权）
新增项目	√	x	x	x	x
编辑项目	√	x	x	x	x
删除项目	√	x	x	x	x
新增服务组	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x
编辑服务组	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x
删除服务组	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（仅自己创建的服务组）	x	x
新增服务	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x
编辑服务	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x
新增服务版本	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x
复制服务	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x
删除服务	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（仅自己创建的服务）	x	x
新增服务依赖	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x	x

操作	工作空间管理员	项目管理员	架构师	开发者	其他（未授权）
删除服务依赖	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（仅自己创建的服务依赖）	x	x
重新生成代码并下载	√（工作空间中所有项目）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	√（工作空间中指定项目内）	x
查看服务详情	√	√	√	√	√
查看变更记录	√	√	√	√	√
申请角色	√	√	√	√	√
撤回申请	√（仅撤回自己的申请）	√（仅撤回自己的申请）	√（仅撤回自己的申请）	√（仅撤回自己的申请）	√（仅撤回自己的申请）
审批角色	√（工作空间中所有项目）	√（工作空间中指定项目内）	x	x	x
删除角色	√（工作空间中所有项目）	√（工作空间中指定项目内）	x	x	x

3.2.4.2 为用户添加工作空间级角色

使用说明

工作空间管理员为对本工作空间下的所有资源具有增删改查的权限，可以为本账号下用户分配项目下的角色。

前提条件

- 只有具备工作空间管理员权限的用户，才能为其他用户添加工作空间级角色。
- 待添加的IAM用户已创建（如IAM用户1）。如何创建IAM用户，请参见[创建IAM用户](#)。

说明

IAM用户（如IAM用户1）必须已添加到用户组admin或group。加入用户组后，用户才会具备用户组的权限。若不加入用户组，登录AstroPro服务时，会提示您没有当前服务的访问权限。

- admin: 缺省用户组，具有所有云服务资源的操作权限。将用户加入该用户组后，用户可以操作并使用所有云服务资源。如果您创建的IAM用户是管理员，才需要将其加入默认用户组“admin”中。
- group: 自定义的用户组，如何创建一个用户组，请参见[创建用户组并授权](#)。自定义用户组时，必须为用户组添加“Astro Pro FullAccess”和“Astro Pro InstanceManagement”系统策略，如[图3-26](#)。

图 3-25 通过用户组为用户授权

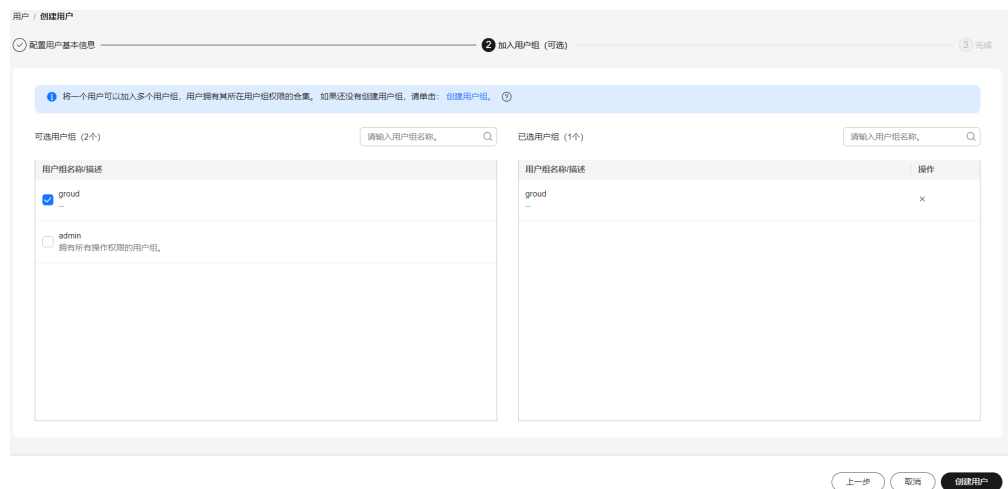


图 3-26 自定义用户组必须具备权限



用户申请角色

步骤1 IAM用户1，登录[华为云网站](#)。

步骤2 在顶部导航栏右侧单击“控制台”，进入华为云控制台。

步骤3 在“产品”中，选择“开发与运维 > 低代码平台 Astro > Astro企业应用 Astro Pro”。

步骤4 在AstroPro服务控制台的首页中，单击实例中的“进入首页”，即可进入AstroPro界面。

首次登录时，请勾选AstroPro隐私协议及服务声明。

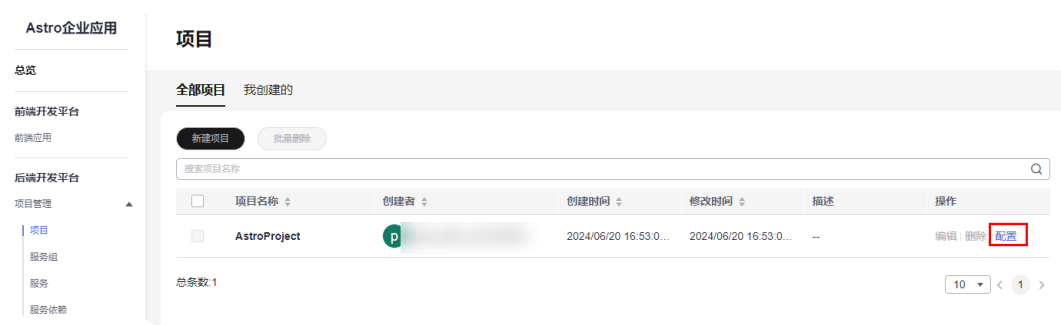
图 3-27 勾选隐私协议及服务声明



步骤5 在左侧导航栏中，选择“项目管理 > 项目”。

步骤6 在项目列表中，单击某个项目后的“配置”。

图 3-28 选择配置



步骤7 在我的申请页签中，单击“新增申请”。

图 3-29 新增申请



步骤8 选择“工作空间管理员”角色，单击“确认”。

图 3-30 选择工作空间管理员




申请完成后，在我的申请列表中，可查看到已申请的记录，且状态为“待审批”。如果需要撤回申请，可单击操作下的 ，撤回已提交的申请。

图 3-31 查看已提交的申请

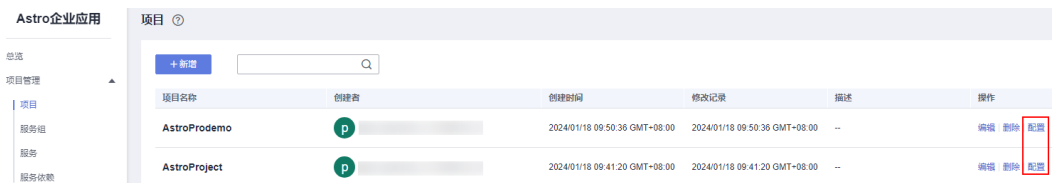


----结束

管理员审批申请

- 步骤1** 工作空间管理员参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 项目”。
- 步骤3** 在项目列表中，单击任意一个项目后的“配置”。

图 3-32 单击任意一个配置按钮



- 步骤4** 在我的待办中，单击“操作”列的“审批”，进入审批页面。

图 3-33 单击审批



- 步骤5** 选择审批结果，输入审批意见，单击“确认”。

图 3-34 审批角色




审批后，在“工作空间角色管理”页签中，可查看到[用户申请角色](#)中用户已具有 workspace_admin 角色。如果需要收回该用户的 workspace_admin 角色，可单击用户后的 ，删除该用户。

图 3-35 查看已添加的用户

用户名称	角色类型	审批人	创建时间	操作
astropro	工作空间管理员		2024/01/18 10:21:14 GMT+08:00	

----结束

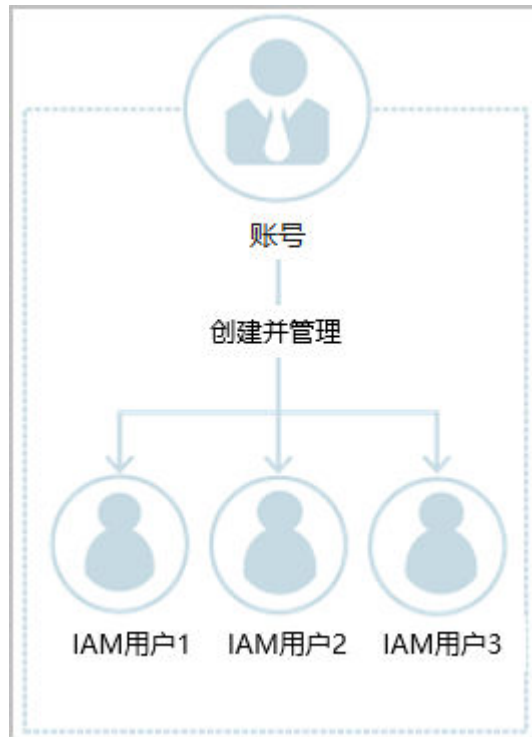
3.2.4.3 为用户添加项目级角色

使用说明

AstroPro 为每个项目预置了项目管理员、架构师和开发者三种类型的角色。

- 项目管理员
 - 可以增删改查项目下的所有资源。
 - 可以为项目下的所有角色分配用户（仅限于同账号下的用户，如[图3-36](#)中的 IAM 用户1、IAM 用户2）。

图 3-36 华为账号与 IAM 用户



- 架构师
 - 可以新建服务。
 - 可以删除本人创建的服务，不能删除其他人创建的服务。
 - 可以对项目下的所有服务进行编辑。
- 开发者
 - 可以查看项目下的所有资源。
 - 可以重新编译有权限项目下的服务并下载代码。

前提条件

- 只有工作空间管理员或项目管理员，才能为其他用户添加项目级角色。不同的是具备工作空间管理员权限的用户，可以为用户添加工作空间下所有项目的权限。而项目管理员只能为用户添加其所在项目的角色权限。
- 待添加的IAM用户已创建（如IAM用户2）。如何创建IAM用户，请参见[创建IAM用户](#)。

📖 说明

IAM用户（如IAM用户1）必须已添加到用户组admin或group。加入用户组后，用户才会具备用户组的权限。若不加入用户组，登录AstroPro服务时，会提示您没有当前服务的访问权限。

- admin: 缺省用户组，具有所有云服务资源的操作权限。将用户加入该用户组后，用户可以操作并使用所有云服务资源。如果您创建的IAM用户是管理员，才需要将其加入默认用户组“admin”中。
- group: 自定义的用户组，如何创建一个用户组，请参见[创建用户组并授权](#)。自定义用户组时，必须为用户组添加“Astro Pro FullAccess”和“Astro Pro InstanceManagement”系统策略，如[图3-38](#)。

图 3-37 通过用户组为用户授权

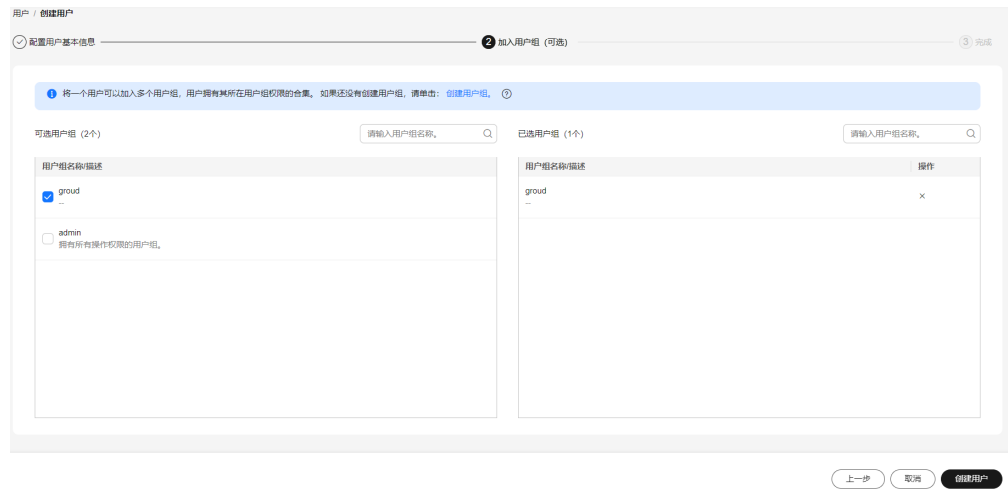


图 3-38 自定义用户组必须具备权限



用户申请角色

- 步骤1** IAM用户2, 登录[华为云网站](#)。
- 步骤2** 在顶部导航栏右侧单击“控制台”, 进入华为云控制台。
- 步骤3** 在“产品”中, 选择“开发与运维 > 低代码平台 Astro > Astro企业应用 Astro Pro”。
- 步骤4** 在AstroPro服务控制台的首页中, 单击实例中的“进入首页”, 即可进入AstroPro界面。

首次登录时, 请勾选AstroPro隐私协议及服务声明。

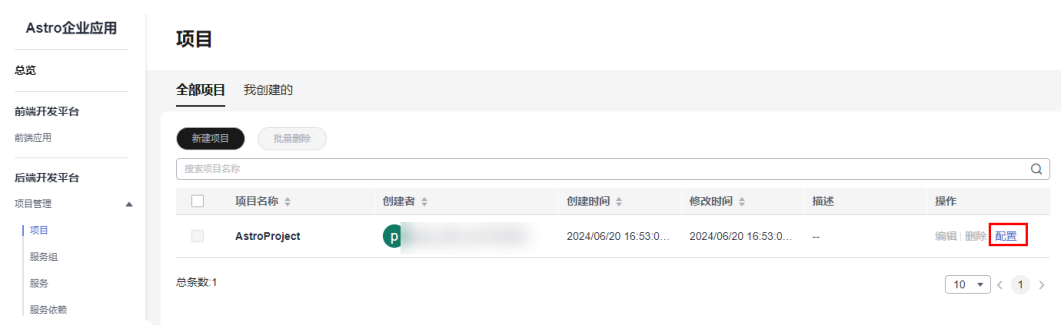
图 3-39 勾选隐私协议及服务声明



步骤5 在左侧导航栏中，选择“项目管理 > 项目”。

步骤6 在项目列表中，单击对应项目后的“配置”。

图 3-40 选择配置



步骤7 在我的申请页签中，单击“新增申请”。

图 3-41 单击新增申请



步骤8 在弹出的申请角色页面中，选择“项目管理员”、“架构师”或“开发者”角色，单击“确认”。

图 3-42 选择项目级角色



申请完成后，在我的申请列表中，可查看到已申请的记录，且状态为“待审批”。如果需要撤回申请，可单击“操作”列的“撤销”，撤回已提交的申请。

图 3-43 查看已提交的申请



----结束

管理员审批申请

- 步骤1** 工作空间管理员或项目管理员参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 项目”。
- 步骤3** 在项目列表中，查找到[用户申请角色](#)操作中，用户申请的项目，单击项目后的“配置”。

说明

如果您是工作空间管理员，可单击任意项目后的“配置”，进行审批。

图 3-44 单击对应项目后的审批



步骤4 在我的待办中，单击“操作”列的“审批”，进入审批页面。

图 3-45 单击审批图标

申请人	工作空间角色类型	项目名称	项目角色类型	创建时间	操作
pro2	--	AstroProject	架构师	2024/06/21 09:38:11 GMT+08:00	审批
pro3	--	AstroProject	开发者	2024/06/21 09:39:11 GMT+08:00	审批
pro1	工作空间管理员	--	--	2024/06/21 09:45:56 GMT+08:00	审批

步骤5 选择审批结果，输入审批意见，单击“确认”。

图 3-46 审批角色

审批角色

审批 同意 拒绝

审批意见: 同意

确认 取消


审批后，在“项目角色管理”页签中，可查看到[用户申请角色](#)中用户已具有所申请的角色。如果需要收回该用户的项目级角色，可单击用户后的，删除该用户。

图 3-47 查看已添加的用户

用户名称	角色类型	审批人	审批时间	操作
pro2	架构师	pro2	2024/06/21 09:55:57 GMT+08:00	删除
pro3	开发者	pro3	2024/06/21 09:49:51 GMT+08:00	删除
pro1	项目管理员	pro1	2024/06/21 09:39:49 GMT+08:00	删除

----结束

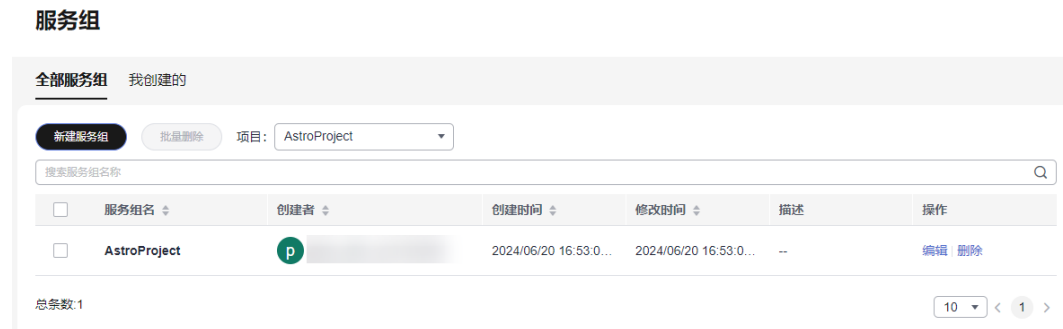
3.3 服务组管理

3.3.1 新建服务组

使用说明

服务组用于对项目中的服务进行分组，一般一个分组对应一个研发团队。创建项目时，会自动创建一个和项目同名的服务组，所有新建服务默认在此分组下。您也可以不使用默认的服务组，直接新建一个服务组。

图 3-48 和项目同名的服务组



前提条件

已参考[新建项目](#)中操作，创建项目。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 服务组”。
- 步骤3** 在项目后的下拉框中，选择[新建项目](#)中创建的项目，单击“新建服务组”。
- 步骤4** 设置服务组的基本信息，单击“确定”。

图 3-49 设置服务组基本信息

< 新建服务组

基本信息

* 服务组名

描述

- 服务组名称：设置新建服务组的名称，只能包含大小写字母、数字、连字符（-）和下划线（_）。
- 描述：输入服务组的描述信息，通常设置为服务组的用途或者功能。

----结束

3.3.2 编辑服务组

使用说明

服务组创建后，支持再次修改服务组的名称和描述。

操作步骤

- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“项目管理 > 服务组”。
- 步骤3 选择服务组所属的项目，单击[新建服务组](#)中已创建服务组后的“编辑”。
- 步骤4 修改服务组的名称和描述，单击“确定”。

----结束

3.3.3 删除服务组

使用说明

服务组不再使用时，可以删除已创建的服务组。删除服务组前，请确保服务组中的服务已删除。如何删除服务，请参见[删除服务](#)。

单个删除服务组

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务组”。

步骤3 选择服务组所属的项目，单击待删除服务组后的“删除”。

步骤4 在弹出的确认框中，单击“确认”，即可删除服务组。

服务组删除后不可恢复，请谨慎操作。

----结束

批量删除服务组

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务组”。

步骤3 选择服务组所属的项目。

步骤4 勾选待删除的服务组，单击“批量删除”。

步骤5 在弹出的确认框中，单击“确认”，即可批量删除服务组。

服务组删除后不可恢复，请谨慎操作。

----结束

3.4 服务管理

3.4.1 了解服务创建流程

什么是（微）服务

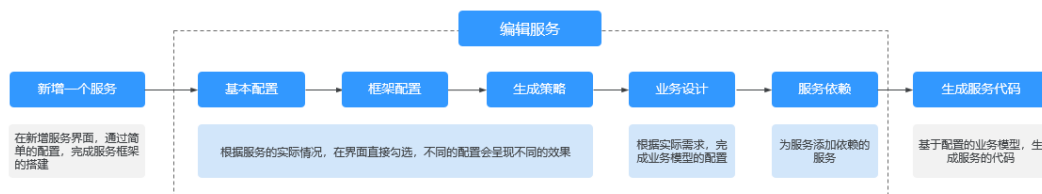
（微）服务是业务概念，即提供某种服务的某个进程。每一个服务都具有自主运行的业务功能，对外开放不受语言限制的API，多个（微）服务组成应用程序。

AstroPro是一个企业应用一站式构建平台，对于平台来说不需要过多的区分服务和微服务。

了解服务创建流程

在AstroPro中创建一个服务的流程，如[图3-50](#)所示。

图 3-50 创建服务流程图



1. 新增一个服务

创建一个空服务，并指定服务的版本。创建服务前，请确保已创建项目和服务组。

2. 基本配置

设置服务框架、版本和单元化策略等信息，请根据实际业务直接在界面进行勾选。

3. 框架配置

设置服务的架构、数据库、缓存和安全认证等信息，请根据实际业务直接在界面进行勾选。

4. 生成策略

设置服务的API、代码风格、部署和性能测试等信息，请根据实际业务直接在界面进行勾选。

5. 业务设计

基本配置、框架配置和生成策略需要用户根据自身业务的实际情况进行配置，配置不同生成的效果有所不同。业务设计是AstroPro的核心能力，是用户设计自己业务的基础。

6. 服务依赖

通常情况下，一个应用不是一个单独的服务，可能由多个服务共同组成。这些服务之间可能存在一些跨服务的调用，此时就需要通过添加依赖服务，把这些服务的客户端集成过来。

7. 生成服务代码

根据配置的业务模型，生成服务的代码。

3.4.2 新增一个服务

使用说明

创建服务前，请确保已创建项目和服务组。如果待添加的服务有依赖其他服务，请先添加依赖的服务。添加被依赖服务时，必须开启“是否生成客户端”。

图 3-51 开启“是否生成客户端”



前提条件

- 参考[新建项目](#)中操作，完成项目的创建。
- 参考[新建服务组](#)中操作，完成服务组的创建。

操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务”。

步骤3 选择项目和服务组，单击“新建服务”。

图 3-52 新增一个服务



步骤4 设置服务的基本信息。

图 3-53 设置服务基本信息

< 新建服务

服务基本信息

* 服务名称

* 服务类型

* 服务组

* 服务单元化策略

* API版本

描述

- 服务名称：设置待添加服务的名称，由英文字母、数字或“-”组成，且必须以字母开头，一般采用驼峰格式，长度最低为两位。
- 服务类型：当前仅支持创建原子服务。原子服务是指对外提供业务对象管理API，有独立数据存储（一般为独立数据库）的服务。原子服务之间可以相互调用。
- 服务组：选择服务所属的分组，即（可选）[步骤2：创建服务组](#)中创建的服务组。
- 服务单元化策略：服务在子域内的单元化策略。服务单元化策略必须在一个子域内定义，不能跨子域。
当前仅支持SINGLE，即单库，无论子域是否进行单元化部署，该服务只在一个单元（一般以region为单元）内部署。
- API版本：指定服务的API版本，对应服务Service段的apiVersion字段，一般为v1、v2类型的值。
- 描述：设置服务的描述信息。

步骤5 单击“确定”，即可完成服务的创建。

此处创建服务的操作，相当于为服务搭建了一个框架。您需要根据自身业务的需求，参考[编辑服务](#)中操作进行业务模型的配置，从而定制出符合您预期的效果。

----结束

3.4.3 编辑服务

3.4.3.1 步骤 1：基本配置

基本信息中配置的内容会呈现在代码中，需用户根据实际情况进行勾选配置。

步骤1 在服务列表中，单击[新增一个服务](#)中已创建服务后的“编辑”。

步骤2 在基本配置中，按需进行设置。

图 3-54 基本配置

- **基本配置**：若本地已有配置好的服务元数据，可通过单击“导入元数据”，直接导入。
- **微服务名称**：自动关联[新增一个服务](#)中创建的服务名称。
- **Group ID**：服务所属项目中的组ID，会自动关联[新建项目](#)中Group的值。在Maven项目中用作工程组的标识，Group ID在一个组织或项目中通常是唯一的。
- **Package**：设置生成代码的顶层包名，会自动关联[新建项目](#)中Package的值。
- **Artifact ID**：在Maven项目中用作工程的标识，通常是工程的名称。
- **版本**：在Maven项目中用作工程的版本号。
- **框架**：选择微服务使用的开发框架，支持DEVSPORE(JDK 8 + SpringBoot 2)和DEVSPORE(JDK 17 + SpringBoot 3)。
 - DEVSPORE(JDK 8 + SpringBoot 2)：生成JDK8+SpringBoot2的代码框架。
 - DEVSPORE(JDK 17 + SpringBoot 3)：生成JDK17+SpringBoot3的代码框架。

步骤3 在详细配置中，配置服务的详细信息。

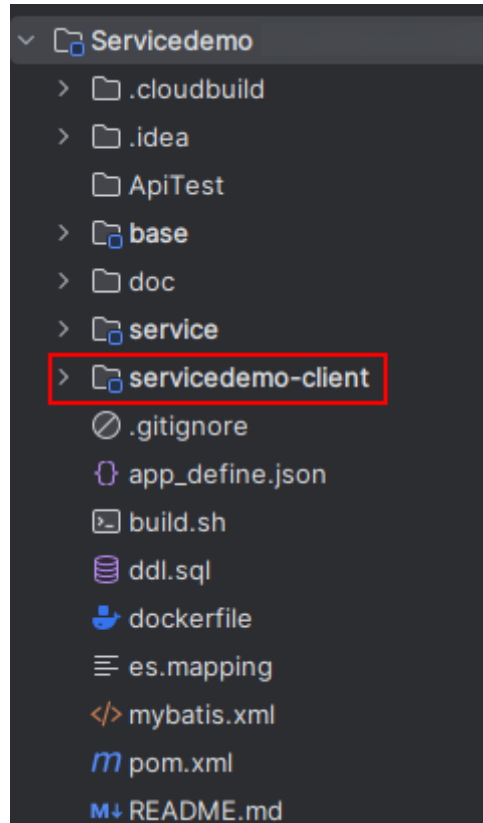
图 3-55 详细配置

- **服务类型**：当前仅支持创建原子服务。原子服务是指对外提供业务对象管理API，有独立数据存储（一般为独立数据库）的服务。原子服务之间可以相互调用。
- **服务组**：选择服务所属的分组。如何创建服务组，请参见[新建服务组](#)。
- **服务单元化策略**：服务在子域内的单元化策略。服务单元化策略必须在一个子域内定义，不能跨子域。

当前仅支持SINGLE，即单库，无论子域是否进行单元化部署，该服务只在一个单元（一般以region为单元）内部署。

- API版本：服务的API版本，默认为**新增一个服务**时配置的版本，如果需要升级API的版本，请参见**升级API版本**。
- 是否生成客户端：是否生成客户端的代码。开启后，会生成服务的客户端代码，如图3-56。

图 3-56 生成客户端的代码



- 是否启用扩展拦截：通过引入devspore-horizon插件，用户自定义继承抽象类 Approve和添加配置，在请求进入和返回时增强处理。
 设置为“是”时，自动在pom文件中引入devspore-horizon插件，并在所有 service实现类的方法上添加“@Extension”注解。同时在plugin目录下，生成“DefaultRequestPlugin.java”示例文件。
 使用插件时，用户需要在配置文件中添加devspore.horizon.processors，即配置自己编写的扩展插件注入到spring中的名称，多个插件之间使用英文逗号隔开，扩展插件需要继承com.huawei.devspore.horizon.approver.Approve抽象类，并重写其中的doApprove方法。

步骤4 设置完成后，单击“下一步”，进入框架配置页面。

----结束

3.4.3.2 步骤 2：框架配置

框架配置和基本配置一样，需要您根据实际情况进行勾选配置，不同的配置会呈现不同的效果。

步骤1 选择参考框架。

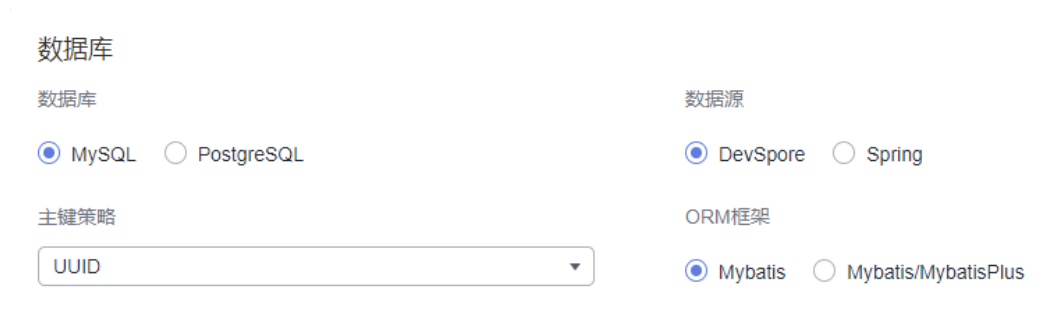
图 3-57 选择参考架构



- 单数据库DB(参考单表查询QPS < 1w)：只有一个数据源。
- 读写分离数据库(参考单表查询QPS < 3w)：默认添加两个数据源，一个负责写数据，一个负责查数据。如果数据源使用DevSpore，就使用devspore-datasource的读写分离模式。如果使用的是Spring数据库，则使用shardingsphere的读写分离模式。
- 读写分离数据库+分布式缓存(参考单表查询QPS < 5w)：数据库开启读写分离，并且开启Redis缓存。
- 数据库分库+分布式缓存(参考单表查询QPS < 5w)：支持数据库分库，分库数量及规则自定义，并且开启Redis缓存。

步骤2 数据库设置。

图 3-58 设置数据库

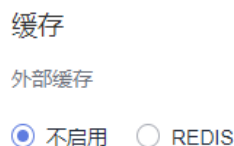


- 数据库：选择数据库的类型，支持MySQL和PostgreSQL。
- 分库策略：选择数据的分片算法。参考架构选择“数据库分库+分布式缓存”时，需要设置。
 - MOD：直接使用分片数取模，余数为分片编号（从0开始编号）。适用整数类型的字段。

- HASH_MOD: 先使用哈希算法, 再使用MOD算法。适用字符串类型的字段。
- RANGE: 按照固定的字段值范围映射到分片编号。适用整数、时间类型的字段。
- CUSTOM: 用户插件实现特定的SPI。适用所有类型的字段。
- INTERVAL: 按照时间间隔分表, 分片列必须为时间类型或时间格式的字符串。
- 分库数量: 设置分库的数量。参考架构选择“数据库分库+分布式缓存”时, 需要设置。
- 分库字段: 设置分库的字段名, 可单击“添加字段”, 按需进行添加。分库对象默认使用根对象主键分库, 根对象默认使用自身主键分库。参考架构选择“数据库分库+分布式缓存”时, 需要设置。
- 主键策略: 设置主键的生成方法。数据库中的主键, 用于唯一标识一条记录。
 - UUID: 使用mybatis interceptor生成的字符串UUID, 分表采用hash, 逻辑表数量难扩容。
 - 雪花算法: 使用ShardingJDBC雪花算法, id以时间戳开头, 分表采用hash, 逻辑表数量难扩容。
 - 自增主键(32位)/自增主键(64位): 使用整数range分表, 需自己开发插件完成分表算法, 逻辑表数量比较容易扩容。
 - 用户自定义: 使用用户自定义的方法。
- 数据源: 设置数据库的SDK类型。
 - DevSpore: DevSpore数据源。
 - Spring: 原生Spring数据源。
- ORM框架: ORM (Object Relational Mapping) 框架采用元数据来描述对象与关系映射的细节, 元数据一般采用XML格式, 并且存放在专门的对象-映射文件中。
 - MyBatis: MyBatis是一款持久化架构, 支持自定义SQL、存储过程和高级映射。MyBatis消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis可以使用简单的XML或注解用于配置和原始映射, 将接口和Java的POJOs (Plain Ordinary Java Objects, 普通的Java对象) 映射成数据库中的记录。
 - MyBatis/MyBatisPlus: MyBatis-Plus是一个MyBatis的增强工具, 为MyBatis提供了一些高效、实用、开箱即用的特性, 使用MyBatis-Plus可以有效的节省开发时间。

步骤3 缓存设置。

图 3-59 缓存设置



- 不启用: 不对接缓存服务。

- REDIS: Redis是一种支持Key-Value等多种数据结构的存储系统, 可用于缓存、事件发布或订阅、高速队列等场景。使用华为云DCS Redis作为缓存中间件, 可简化缓存参数维护操作。更多关于DCS Redis的介绍, 请参见[分布式缓存服务DCS](#)。

步骤4 安全认证设置。

图 3-60 安全认证设置

安全认证

身份认证 密码加密

不启用 华为云OneAccess 不启用 开源Jasypt

参数校验

不启用 Hibernate

- 身份认证
 - 不启用: 不启用安全认证机制。
 - 华为云OneAccess: 使用OneAccess作为安全认证机制。华为云OneAccess是一个贯穿企业全业务流程的身份安全管理服务。更多关于OneAccess的介绍, 请参见[应用身份管理服务OneAccess](#)。
- 密码加密: 配置文件中密码加解密方式。
 - 不启用: 不内置加解密方式。
 - 开源Jasypt: 使用开源Jasypt进行加解密。
- 参数校验: 参数校验使用的类型。
 - 不启用: 不对参数进行校验。
 - Hibernate: 使用Hibernate注解参数校验方式。

步骤5 云服务设置。

图 3-61 云服务设置

云服务

注册发现/配置中心 调用链

不启用 CSE NACOS 不启用 华为云APM

服务监控

不启用 华为云AOM

- 注册发现/配置中心
 - 不启用: 不对接配置管理服务。
 - CSE: 使用微服务引擎服务CSE作为配置管理服务。CSE是微服务应用的云中间件, 为用户提供了注册发现、服务治理、配置管理等高性能和高韧性的企

业级云服务能力，可无缝兼容Spring Cloud、ServiceComb等开源生态，用户也可以结合其他云服务，快速构建云原生微服务体系，实现微服务应用的快速开发和高可用运维。更多关于CSE的介绍，请参见[微服务引擎CSE](#)。

- NACOS: 使用NACOS作为配置管理服务。NACOS提供了一组简单易用的特性集，帮助您快速实现动态服务发现、服务配置、服务元数据及流量管理。
- 调用链
 - 不启用: 不启用调用链。
 - 华为云APM: 使用应用性能管理服务APM作为调用链。APM您的云上引用健康管理专家，可帮助运维人员快速发现应用的性能瓶颈，以及故障根源的快速定位，为用户体验保驾护航。更多关于APM的介绍，请参见[应用性能管理APM](#)。
- 服务监控
 - 不启用: 不对接服务监控组件。
 - 华为云AOM: 使用应用运维管理服务AOM作为服务监控组件。应用运维管理AOM是云上应用的一站式立体化运维管理平台，实时监控您的应用及相关云资源，分析应用健康状态，提供灵活丰富的数据可视化功能，帮助您及时发现故障，全面掌握应用、资源及业务的实时运行状况。更多关于AOM的介绍，请参见[应用运维管理AOM](#)。

步骤6 设置完成后，单击“下一步”，进入生成策略页面。

----结束

3.4.3.3 步骤 3：生成策略

生成策略和基本配置、框架配置一样，需要您根据实际情况进行勾选配置，不同的配置会呈现不同的效果。

步骤1 API设置。

图 3-62 API 设置

API

Json序列化

小驼峰 下划线

Web框架模型

WebMVC

标准响应体

是 否

关闭API中根对象前缀

是 否

查询参数阈值

- Json序列化
 - 小驼峰: 序列化后的json属性名，采用驼峰格式。
 - 下划线: 序列化后的json属性名，采用下划线连接单词。
- 标准响应体: 返回的响应体是否使用标准样式。


```
{
  "code": 200,
  "msg": "success",
  "data": {
    "name": "zhangsan",
    "birthday": "1990-01-01",
    "other_properties": "..."
  }
}
```

- Web框架模型：生成基于spring-webmvc的API层。
- 关闭API中根对象前缀：设置为“是”时，sharding bo的API前面不需要添加root bo的路径。
- 查询参数阈值：设置查询参数阈值，值为“0”时不生效。当查询参数大于该阈值时，将多个查询参数转换为对象。

步骤2 设置代码风格。

图 3-63 设置代码风格

代码风格

配置文件格式

yaml properties

Lombok插件

是 否

工程目录

base/service 单Module DDD

module添加服务名前缀

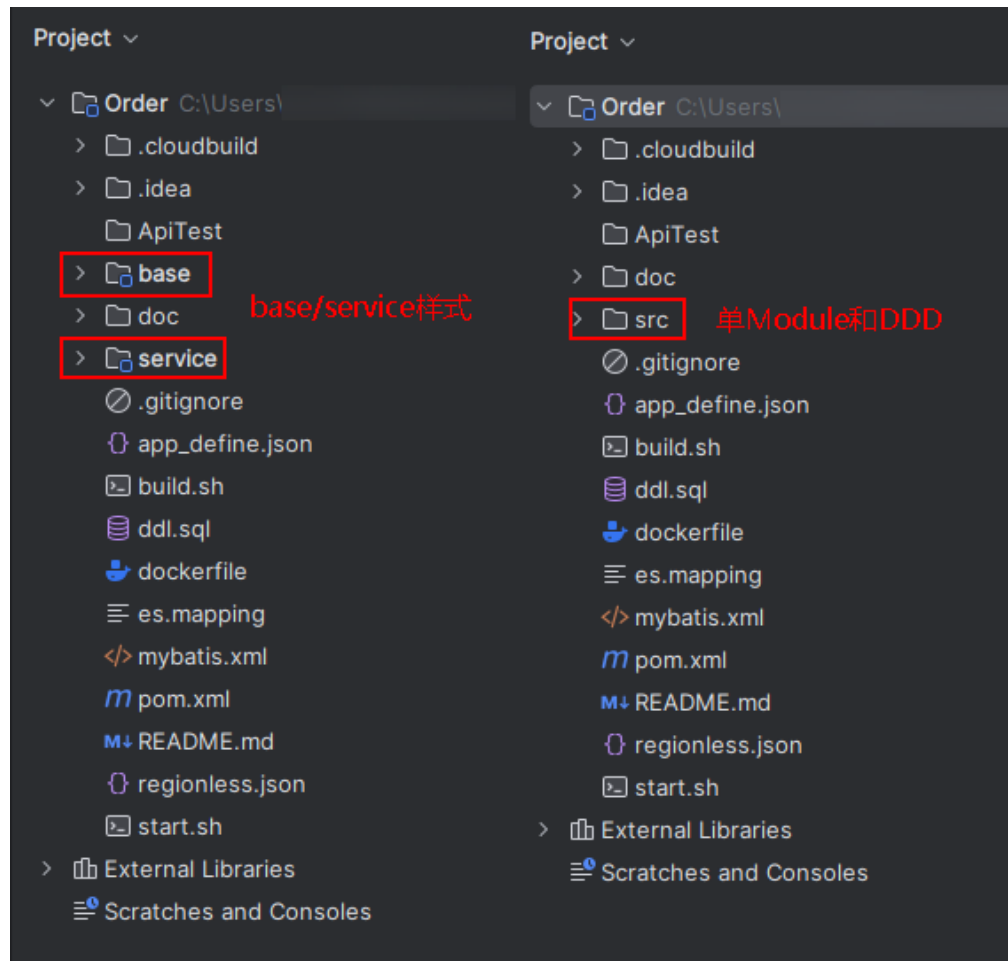
是 否

是否关闭注解

是 否

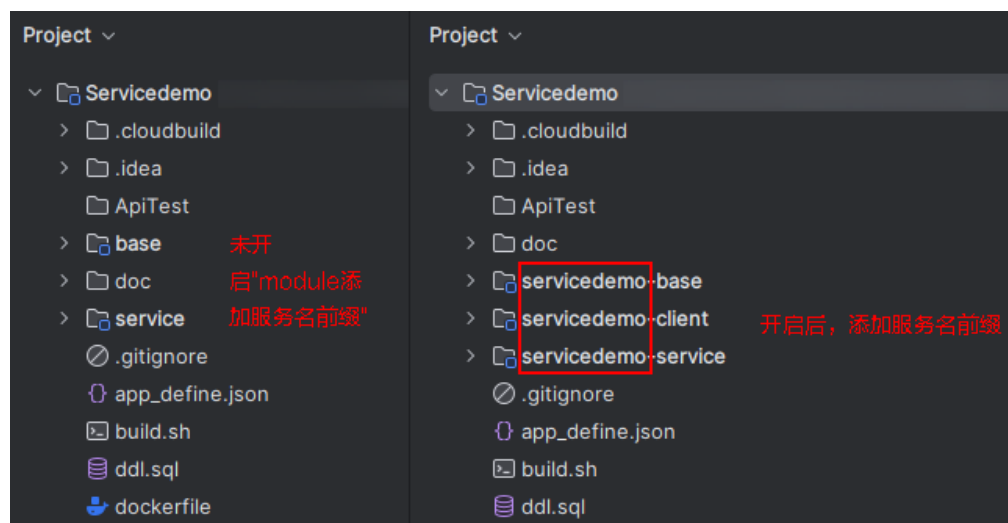
- 配置文件格式：配置spring boot properties文件格式。
 - yaml：配置文件使用yaml格式。
 - properties：配置文件使用properties格式。
- Lombok插件：是否为DO、DTO或QO定义类自动生成Lombok注解。
- 工程目录：设置生成代码的工程目录样式。
 - 单Module：工程目录结构只有一个模块。
 - base/service：工程目录结构包含base和service两个模块。
 - DDD：和单Module一样，工程目录结构只有一个模块。

图 3-64 工程目录不同类型设置效果



- module添加服务名前缀：配置为“是”时，模块名称前会添加服务名前缀。

图 3-65 开启前后效果



步骤3 设置部署信息。

图 3-66 设置部署信息

部署

服务部署脚本

CCE ServiceStage 否

服务打包方式

jar包 war包

- 服务部署脚本
对接CCE部署和服务Stage部署时，生成的代码中会包含如下内容：
 - 根目录中会增加 “.cam” 文件夹，包含 “cam.yml” 和 “variables.yml” 文件。
 - service模块的 “application.yam” 文件中，会增加 “server.tomcat” 配置参数。
 - dockerfile脚本会做相应的修改。
- 服务打包方式
 - jar：打成jar包。jar通常包含一些Java类文件、相关元数据和资源，在声明了Main_class后可使用java命令运行。
 - war：打成war包。war是Java Web应用程序的标准打包格式，war是一个Web模块，包括WEB-INF目录，可直接运行于Web容器中。

步骤4 性能测试。

图 3-67 性能测试

测试

CodeArts性能测试

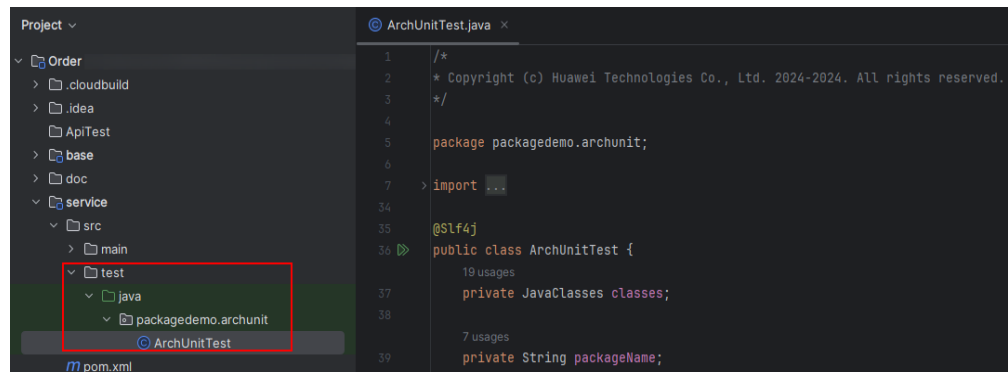
是 否

代码架构看护测试

是 否

- CodeArts性能测试：是否使用CodeArts PerfTest进行性能测试。性能测试CodeArts PerfTest是一项为基于HTTP/HTTPS/TCP/UDP/HLS/RTMP/WEBSOCKET/HTTP-FLV等协议构建的云应用提供性能测试的服务，支持快速模拟大规模并发用户的业务高峰场景，可以很好的支持报文内容和时序自定义、多事务组合的复杂场景测试，测试完成后会为您提供专业的测试报告呈现您的服务质量。更多关于CodeArts PerfTest的介绍，请参见[性能测试CodeArts PerfTest](#)。
- 代码架构看护测试：是否启用看护代码架构，看护代码的分层调用、命名规范和注解规范等。“代码架构看护测试”设置为“是”时，在代码中会生成一个“test”目录。

图 3-68 开启代码结构看护测试效果



步骤5 租户配置。

图 3-69 租户配置



- 多租模型：是否支持多租户资源隔离。
 - Tenant：支持多租模式，BO级多租配置multiTenant生效，开启BO级多租的业务对象必须关联到租户，包含租户id字段，租户id字段可自定义名称。
 - 否：无内置租户模型。
- 租户验证方式：设置租户ID资源获取方式。“多租模型”设置为“Tenant”时，才显示该配置。
 - Header：使用header头携带方式传入tenantId。
 - Token：使用token方式传入tenantId。

步骤6 设置完成后，单击“下一步”，进入业务设计页面。

----结束

3.4.3.4 步骤 4：业务设计

步骤1：基本配置、**步骤2：框架配置**和**步骤3：生成策略**中参数，只需要用户根据自身业务直接在界面进行勾选配置。而业务设计需要用户根据实际的需求，进行业务模型的设计和配置。

例如，创建一个简单的订单系统，订单系统中包括用户（User）、订单（Order）和订单详情（OrderDetail）三个业务对象，且三个对象之间存在聚合关系，即用户存在时，订单才会存在，订单存在时，订单详情才会存在。同时一个用户可以关联多个订单，订单通过单号进行标识，一个订单又可以关联多个商品，例如手机、耳机等，商品可以记录数量。实现上述业务逻辑，需要进行如下设计：

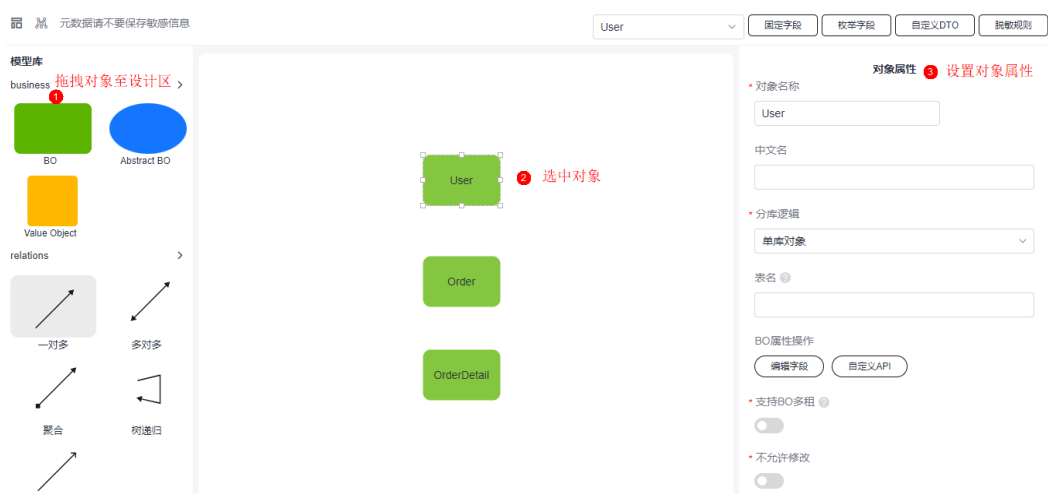
步骤1 在业务设计页面，拖拽所需的对象到设计区，并修改对象名称。

AstroPro提供了BO、Abstract BO和Value Object三种类型的对象，请根据业务需求进行选择。

- **BO**：业务对象，业务对象映射到服务中的一个实体，对应数据库中的一张表。
- **Abstract BO**：抽象对象，不能实例化，没有对应的数据库表，需要和业务对象有个继承的操作。例如，业务对象A继承一个抽象对象B，则B中的字段都会被A继承过来。
- **Value Object**：值对象，不能单独存在，需要和业务对象建立聚合的关系。

本示例中，拖拽三个BO对象到设计区，选中对应的BO，修改对象名称为User、Order和OrderDetail。

图 3-70 拖拽三个 BO 到设计区



步骤2 设置对象属性。

本示例中，因为一个用户需要关联多个订单，订单通过单号进行标识，一个订单又可以关联多个商品，商品可以记录数量。所以需要为“User”添加“name（用户名）”字段，用于记录用户信息。为“Order”添加“orderNo（订单编号）”字段，用于记录订单的编号。为“OrderDetail”添加“product（商品）”、“amount（数量，integer类型）”字段，分别用于记录商品的详情和商品的数量。

图 3-71 为 User 对象添加 name

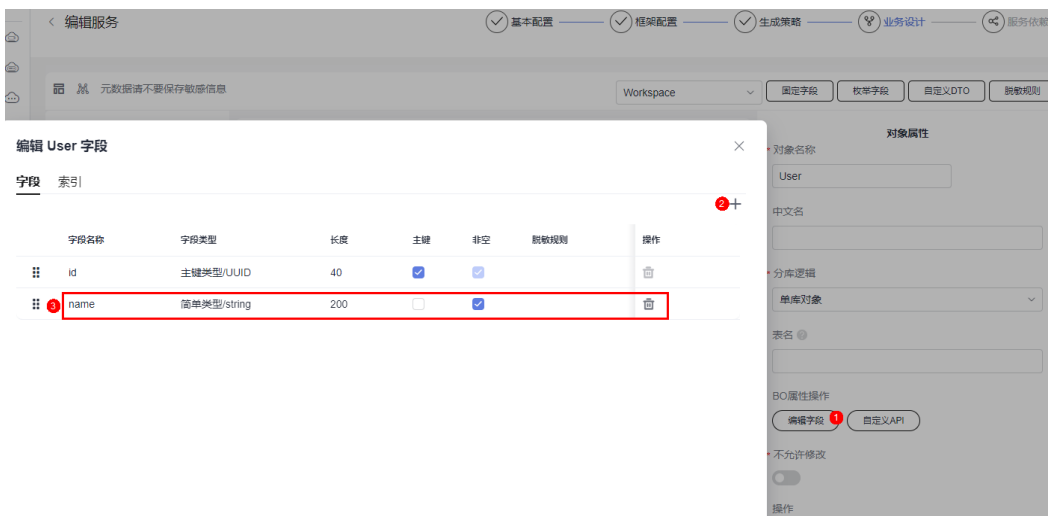


图 3-72 为 Order 添加 orderNo

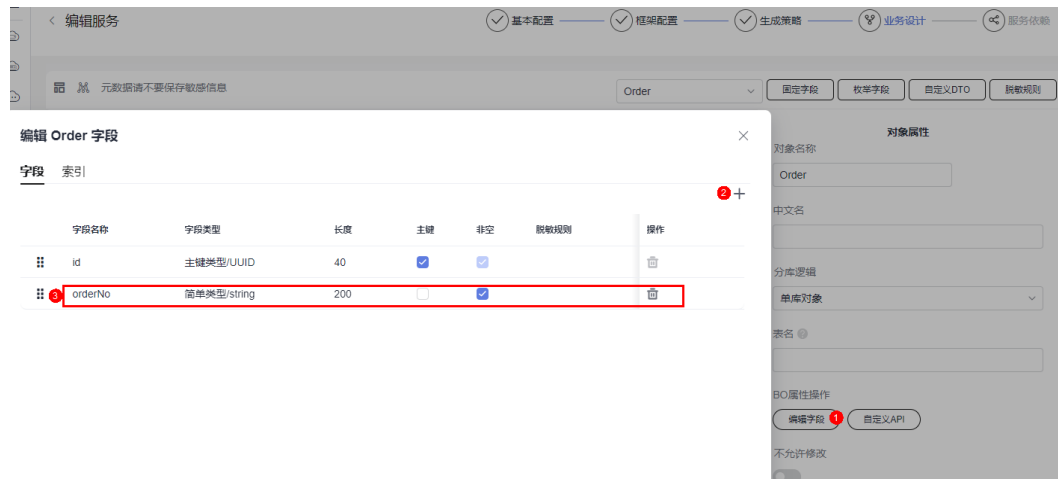
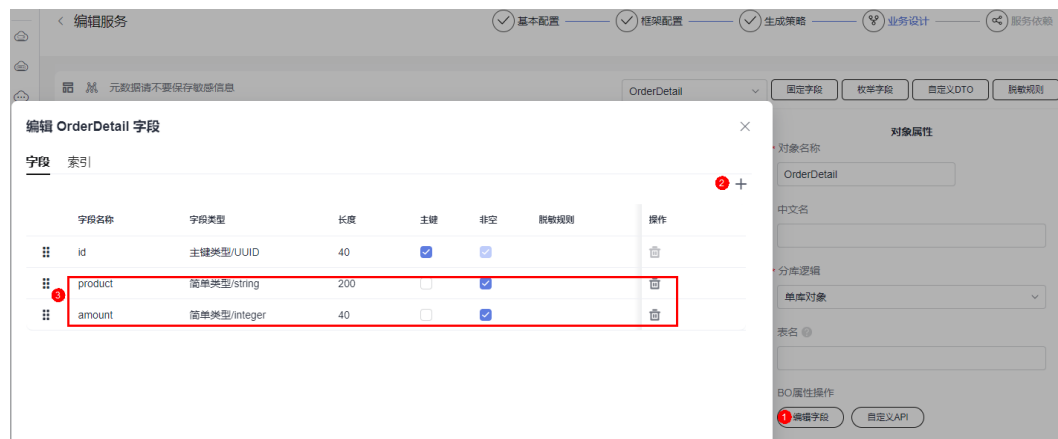


图 3-73 OrderDetail 添加 product 和 amount

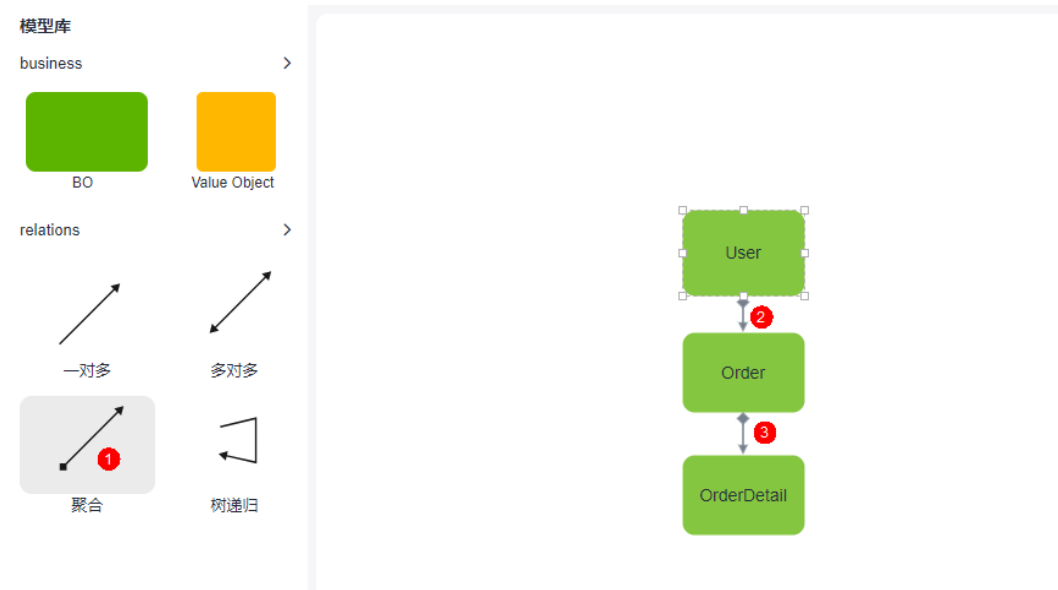


步骤3 建立对象之间的关系。

AstroPro提供了一对多、多对多、聚合、树递归和继承五大关系，请根据自身业务需求进行选择。各对象间关系详细介绍，请参见[对象间关系](#)。

本示例中的订单系统，当用户存在时，订单才会存在，订单存在时，订单详情才会存在，此时需要为三个对象之间建立聚合关系。聚合关系中，次要方必须依赖首要方，任何对于次要方的操作首先要经过首要方才能继续往下操作。在User和Order的聚合关系中，User为首要方，Order为次要方，即用户存在时，订单才会存在。在Order和OrderDetail的聚合关系中，Order为首要方，OrderDetail为次要方，即订单存在时，订单详情才会存在。

图 3-74 设置对象间关系



步骤4 设置完成后，单击“下一步”，进行服务依赖设置。

----结束

3.4.3.5 步骤 5：服务依赖

通常情况下，一个应用不是一个单独的服务，可能由多个服务共同组成。这些服务之间可能存在一些跨服务的调用，此时就需要通过添加依赖服务，把这些服务的客户端集成过来。添加依赖服务前，请确保依赖服务的“是否生成客户端”按钮已启用。如果此处未添加依赖的服务，服务编辑完成后，可在“项目管理 > 服务依赖”中进行添加，详情请参见[新增依赖服务](#)。

图 3-75 开启“是否生成客户端”配置



步骤1 在服务依赖中，选择当前服务依赖的服务。

图 3-76 选择依赖的服务



步骤2 选择依赖服务的版本号。

步骤3 在依赖强弱中，选择strong（强）或weak（弱），单击“添加”，完成依赖服务的添加。

若此处不添加依赖服务，可在服务创建完成后，参考[新增依赖服务](#)中操作，添加服务依赖关系。

图 3-77 完成依赖服务的添加



----结束

3.4.4 生成服务代码

使用说明

根据配置的业务模型，生成服务的基本代码。代码生成后，会提供一个压缩包，供您使用。关于代码结构的详细介绍，请参见[服务开发框架详解](#)。

操作步骤

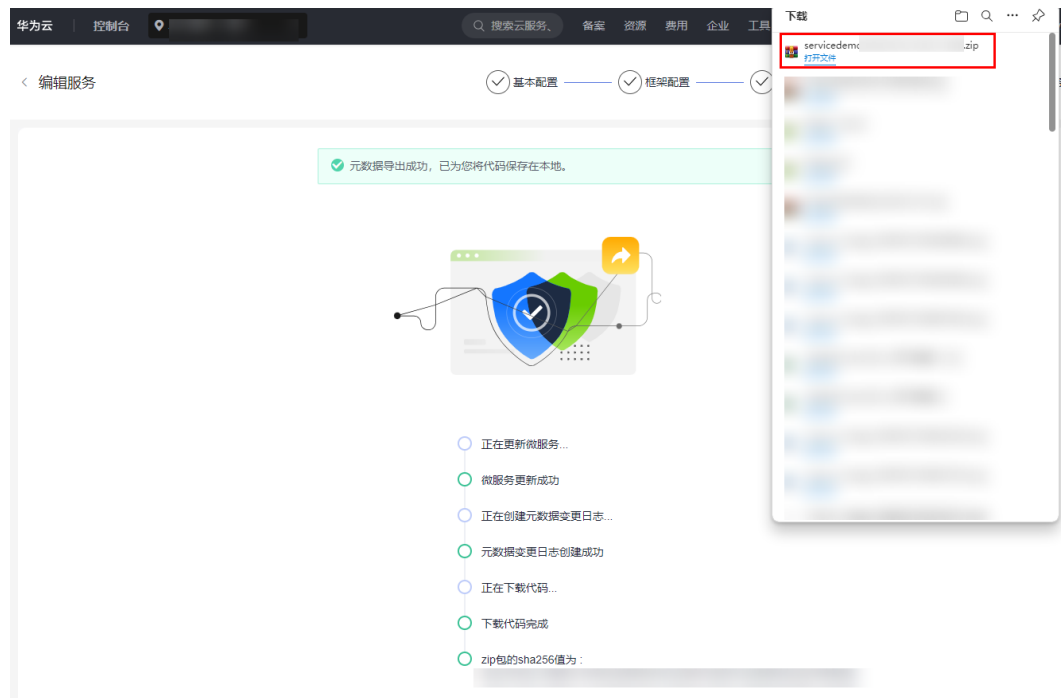
步骤1 参考[编辑服务](#)中操作，完成业务模型的配置。

步骤2 在服务依赖中，单击“创建”。

步骤3 输入变更日志描述信息，单击“创建”。

系统开始创建服务，并生成该服务的代码。

图 3-78 创建服务并生成代码包



----结束

3.4.5 查看服务详情

使用说明

服务创建后，在服务详情中可查看服务的基本信息、变更记录和所依赖的服务等信息，还可以执行编辑服务、重新生成服务代码和导出元数据等操作。

操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务”。

步骤3 在服务列表中，单击已创建服务后的“详情”。

步骤4 在服务详情页，可按需执行相关操作。

- 查看服务的基本信息、变更记录和所依赖的服务等信息。
- 在所属服务组中，单击“编辑”，可修改服务组的名称和描述信息。
- 在服务信息中，单击“编辑”，可对服务进行再次编辑。
- 在服务信息中，单击“重新生成”，可再次生成服务的代码。
- 在服务信息中，单击 \dots ，选择“新增版本”，可修改服务的API版本。
- 在服务信息中，单击 \dots ，选择“删除”，可删除该服务。
- 在变更记录中，单击服务名称后的“导出元数据”，可导出服务的元数据信息。元数据导出后，在创建服务时可直接导入使用。
- 在变更记录中，单击对应服务名称后的“查看”，可查看该服务的信息。单击“重新生成”，可生成该服务的代码。

----结束

3.4.6 升级 API 版本

使用说明

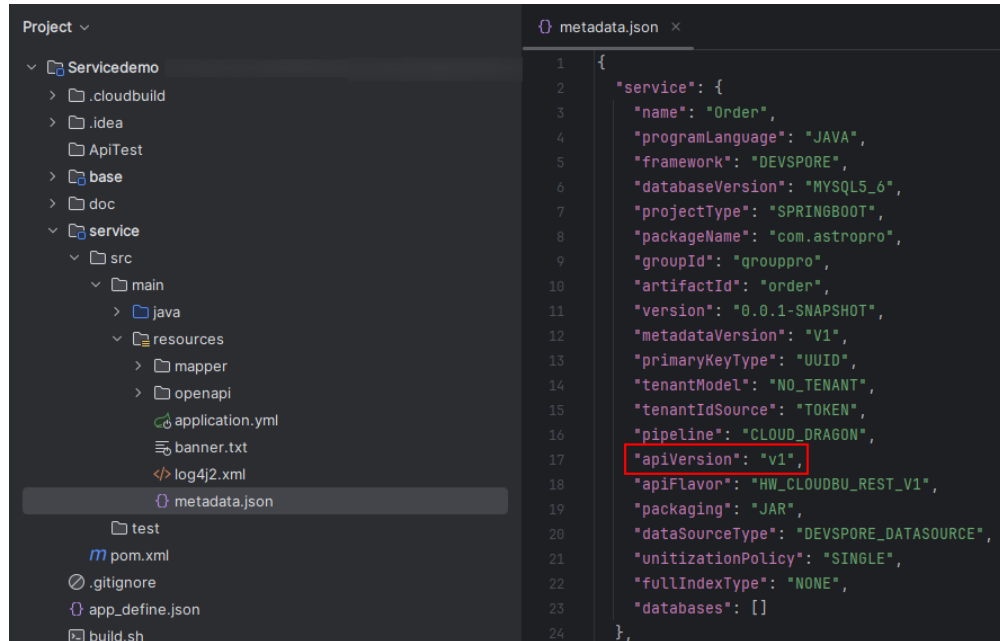
服务创建后，若需要修改服务的API版本号，可通过新增版本实现。

图 3-79 API 版本号

The screenshot displays the configuration page for a service in AstroPro. It is divided into two main sections: 'Basic Configuration' and 'Detailed Configuration'. In the 'Basic Configuration' section, the 'API Version' field is highlighted with a red border and contains the text 'v. 1'. Other fields in this section include 'Service Name' (Order), 'Group ID' (grouppro), 'Package' (com.astropro), 'Artifact ID' (order), 'Version' (0.0.1-SNAPSHOT), and 'Framework' (DEVSPORE(JDK 8 + SpringBoot 2)). The 'Detailed Configuration' section includes 'Service Type' (原子服务), 'Service Group' (AstroProject), 'Service Unit Strategy' (SINGLE), 'Whether to generate client' (checked), 'Client Type' (OPEN_FEIGN), and 'Whether to enable extension column' (unchecked).

API版本对应服务metadata.json文件中，Service段的apiVersion字段，一般为v1、v2类型的值。

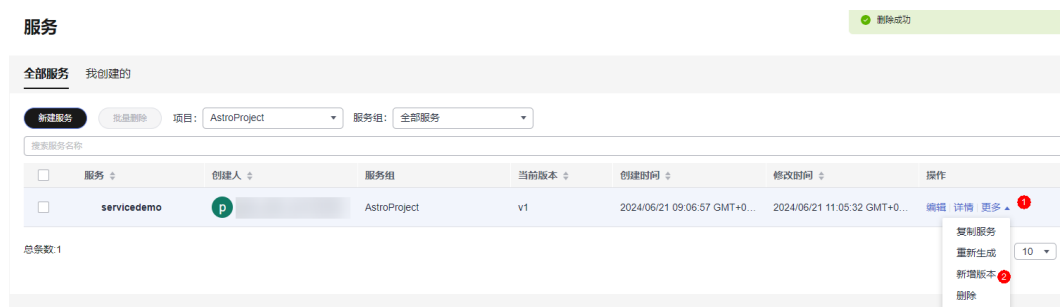
图 3-80 查看 apiVersion 取值



操作步骤

- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“项目管理 > 服务”。
- 步骤3 在服务列表中，选择已创建服务后的“更多 > 新增版本”。

图 3-81 选择新增版本



- 步骤4 在“基本配置 > 详细配置 > API版本”中，修改API的版本号。

图 3-82 修改版本号



步骤5 单击“下一步”，直至生成服务代码。

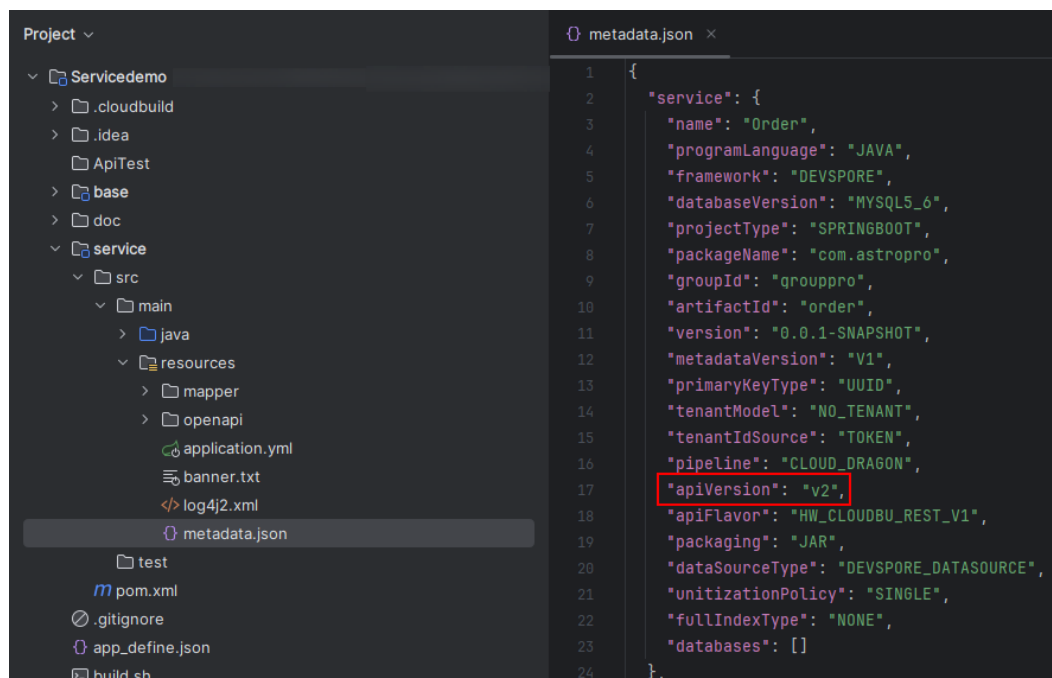
在服务列表中，可查看到服务的当前版本已修改为“v2”。生成服务的metadata.json文件中，Service段的apiVersion字段取值也同步修改为“v2”。

图 3-83 查看修改后的版本号



服务	创建人	服务组	当前版本	创建时间	修改时间	操作
servicedemo		AstroProject	v2	2024/06/21 09:06:57 GMT+0...	2024/06/21 11:15:33 GMT+0...	编辑 详情 更多
servicedemo		AstroProject	v1	2024/06/21 09:06:57 GMT+0...	2024/06/21 11:05:55 GMT+0...	编辑 详情 更多

图 3-84 查看修改后的 apiVersion 取值



----结束

3.4.7 重新编译服务

使用说明

服务创建编译完成后，支持再次编译服务并生成新的代码压缩包。

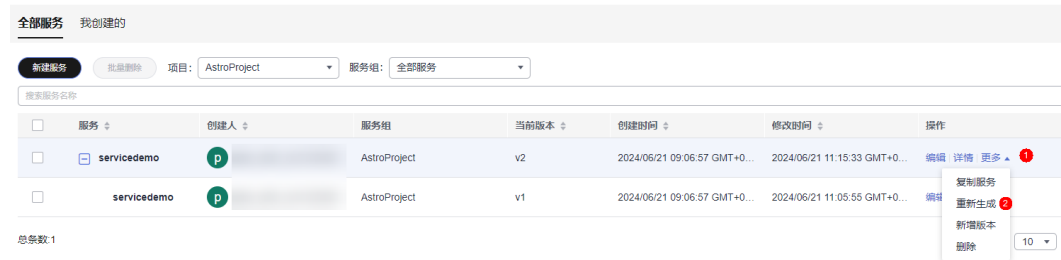
操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务”。

步骤3 在服务列表中，单击已创建服务后的“更多 > 重新生成”，即可重新生成服务的代码。

图 3-85 重新生成服务代码



单击操作中的“详情”，在服务详情页，同样可以对服务进行重新生成。

图 3-86 在详情页重新编译服务



图 3-87 重新生成代码并下载



---结束

3.4.8 复制服务

使用说明

AstroPro支持复制服务，减少重复开发，提高交付效率。

前提条件

仅 workspace 管理员、项目管理员和架构师，才能执行复制服务的操作。

图 3-88 复制服务角色要求



操作步骤

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务”。

步骤3 在服务列表中，单击对应服务后的“更多 > 复制服务”。

复制成功后，自动进入服务编辑页面，您可以直接使用服务，也可以进行二次开发。

图 3-89 选择复制服务



----结束

3.4.9 删除服务

使用说明

当服务不再使用时，可删除已创建的服务。若服务存在依赖服务，请先删除依赖关系，再删除该服务。如何删除服务依赖，请参见[删除服务依赖](#)。

📖 说明

服务删除后不可恢复，请谨慎操作。

单个删除服务

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务”。

步骤3 选择服务所属的项目。

步骤4 在服务列表中，单击已创建服务后的“更多 > 删除”。

步骤5 在弹出的确认框中，单击“确认”，即可删除服务。

当服务有多个版本时，仅会删除对应版本的服务，其他版本的服务不会删除。

图 3-90 删除 v2 版本的服务



----结束

批量删除服务

步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。

步骤2 在左侧导航栏中，选择“项目管理 > 服务”。

步骤3 选择服务组所属的项目。

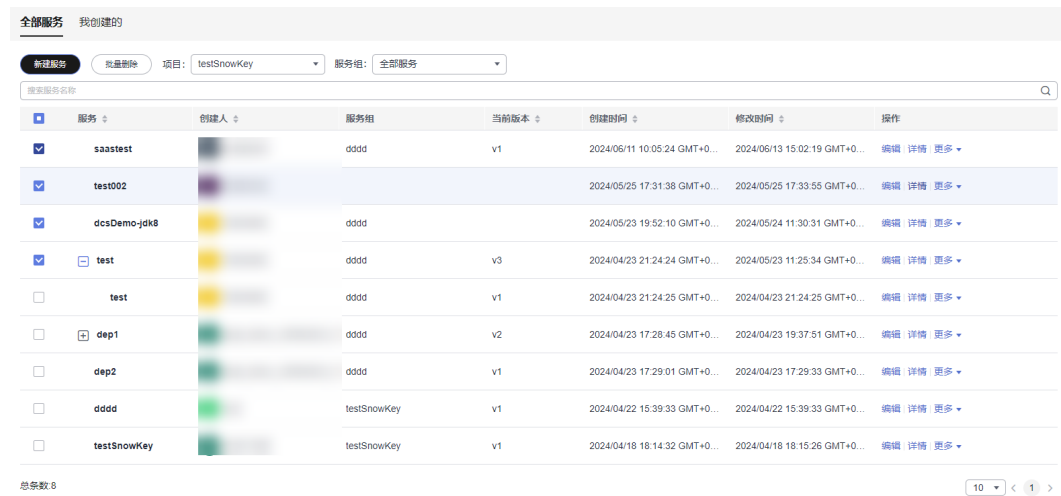
步骤4 在服务列表中，勾选待删除的服务。

步骤5 单击“批量删除”。

步骤6 在弹出的确认框中，单击“确认”，即可批量删除服务。

当服务有多个版本时，仅会删除对应版本的服务，其他版本的服务不会删除。

图 3-91 删除 test 服务 v3 版本



----结束

3.4.10 导出元数据

使用说明

服务创建后，支持导出服务的元数据。元数据导出后，在创建类似服务时，可通过“导入元数据”，直接导入使用。

图 3-92 导入元数据



操作步骤

- 步骤1 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2 在左侧导航栏中，选择“项目管理 > 服务”。
- 步骤3 在服务列表中，单击已创建服务后的“详情”，进入服务详情页。
- 步骤4 在变更记录中，单击“导出元数据”，即可将元数据以json格式导出到本地。

图 3-93 导出元数据



----结束

3.5 服务依赖管理

3.5.1 新增依赖服务

使用说明

通常情况下，一个应用不是一个单独的服务，可能由多个服务共同组成。这些服务之间可能存在一些跨服务的调用，此时就需要通过添加依赖服务，把这些服务的客户端集成过来。如果在[编辑服务](#)中没有添加服务依赖，服务创建后，可在服务依赖中进行添加。

图 3-94 未添加服务依赖

前提条件

- 添加被依赖服务时，请确保被依赖的服务已开启“是否生成客户端”。

图 3-95 开启“是否生成客户端”配置

- 请确保服务和被依赖的服务属于同一项目。例如，服务A依赖服务B，服务A属于AstroProject项目，则服务B也必须要属于AstroProject项目。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
- 步骤2** 在左侧导航栏中，选择“项目管理 > 服务依赖”。
- 步骤3** 在项目下拉框中，选择服务所属的项目，单击“新建依赖服务”。
- 步骤4** 配置服务的基本信息，单击“下一步”。

图 3-96 配置依赖服务基本信息

< 新建服务依赖

基本信息

* 服务

servicedemo

* 服务版本

v2

* 服务依赖

servicedetest

* 依赖版本

v1

- 服务：选择已创建的服务。
- 服务版本：选择服务的版本号。
- 服务依赖：选择被依赖的服务（服务提供方）。
- 依赖版本：选择依赖服务的版本号。

步骤5 设置规则，单击“完成”。

图 3-97 规则定义

< 新建服务依赖

规则定义

* 调用方式

REST

* 客户端类型

OPEN_FEIGN

* 控制器类型

SPRING_WEBMVC

* 依赖强弱

weak

- 调用方式：服务通过API调用被依赖服务的方式，具体的API调用方式取决于被依赖服务。
- 客户端类型：配置客户端远程调用工具类型，默认为OPEN_FEIGN，即spring-cloud:openFeign远程调用客户端。
- 控制器类型：设置生成API层时的生成依据，默认为SPRING_WEBMVC，即生成基于spring-webmvc的API层。
- 依赖强弱：设置依赖关系的类型。
 - strong：被依赖服务不可用，依赖服务调用的被依赖服务的相关API也是不可用状态。
 - weak：被依赖服务不可用，依赖服务的可用性不受影响。

步骤6 单击“完成”，完成服务依赖的创建。

----结束

3.5.2 查看服务依赖

使用说明

服务依赖创建后，只能查看不支持再次编辑。

操作步骤

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
 - 步骤2** 在左侧导航栏中，选择“项目管理 > 服务依赖”。
 - 步骤3** 在项目下拉框中，选择服务所属的项目。
 - 步骤4** 在依赖服务列表中，单击服务后的“查看”，即可查看服务间的依赖。
- 结束

3.5.3 删除服务依赖

使用说明

需要解除服务之间的依赖时，可以通过删除服务依赖来实现。

单个删除服务依赖

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
 - 步骤2** 在左侧导航栏中，选择“项目管理 > 服务依赖”。
 - 步骤3** 选择服务依赖所属的项目，在服务依赖列表中单击操作列的“删除”。
 - 步骤4** 在弹出的确认对话框中，单击“确认”，即可删除服务依赖。
- 结束

批量删除服务依赖

- 步骤1** 参考[登录AstroPro界面](#)中操作，登录AstroPro界面。
 - 步骤2** 在左侧导航栏中，选择“项目管理 > 服务依赖”。
 - 步骤3** 选择服务依赖所属的项目。
 - 步骤4** 在服务依赖列表中，勾选待删除的服务依赖。
 - 步骤5** 单击“批量删除”。
 - 步骤6** 在弹出的确认对话框中，单击“确认”，即可批量删除服务依赖。
- 结束

3.6 对象详解

3.6.1 BO

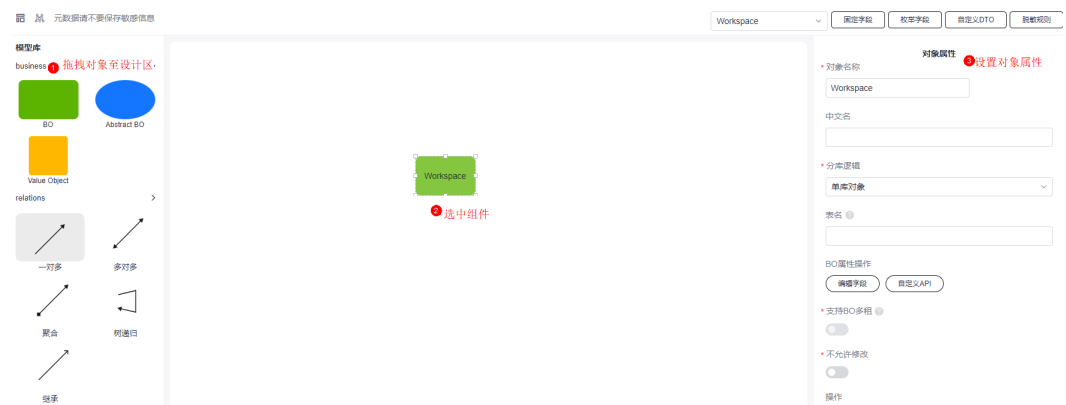
功能介绍

BO (Business Object) 是一个业务对象，业务对象映射到服务中的一个实体，对应数据库中的一张表。

属性说明

在业务设计页面，从“business”中，拖拽“BO”对象至画布空白区域。选中对象，在右侧页面设置对象属性，如图3-98所示。

图 3-98 BO

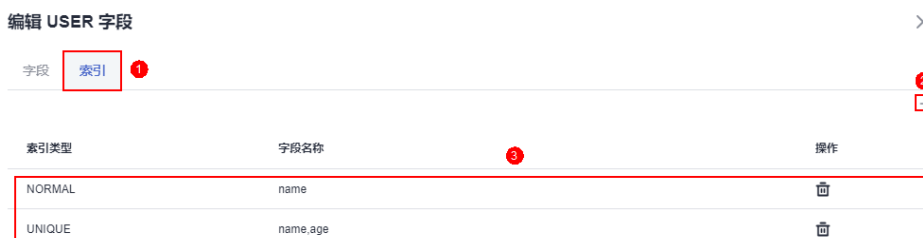


- 对象名称：设置对象的名称，必须使用大驼峰格式，不允许存在连续的大写字母。
- 中文名：设置BO对象的中文名称。
- 分库逻辑：设置BO数据分库（sharding）策略。
 - 广播对象（BROADCASTING）：不需要进行分库，数据在所有数据分库实例上复制，一般是包含配置信息或者维度数据的小表。
 - 根对象（ROOTED）：分库的根对象，每个服务只能有一个。支持的主键类型有UUID和雪花算法，其中雪花算法支持逻辑数据库水平扩展。
 - 分库对象（SHARDING）：需要进行分库，所有对象实例必须有字段关联的ROOTED BO（AstroPro会自动创建ROOTED表引用字段，并创建外键关联ROOTED表），SHARDING表的分库策略和ROOTED必须保持一致。
 - 单库对象（SINGLE）：单表，不进行分库，也不是广播表，仅在一个数据库实例中存在。
- 表名：设置BO对象在数据库中的表名。
- BO属性操作：设置对象的属性操作。
 - 编辑字段：编辑对象的字段。单击“编辑字段”，进入编辑对象字段页面，可为对象添加字段和索引。其中，“NORMAL”为一般类型索引，使用B+树类型存储；“UNIQUE”为唯一索引。

图 3-99 新增字段



图 3-100 添加索引



- 自定义API: 当系统预置的增删改查操作不能满足需求时, 您可以通过自定义API来实现某个操作。如何为对象自定义API, 请参见[如何为对象自定义API](#)。
- 支持BO多租: BO是否支持多租。开启BO多租时, 请确保“多租模型”已开启。

说明

购买AstroPro专业版实例时, 才会显示“转测BO多租”这个配置项。

图 3-101 开通多租模型

租户配置

多租模型 Tenant 否 租户验证方式 Header Token

- 不允许修改: 对象是否支持修改。
- 操作: 对新建的对象可执行哪些操作, 如create (新建)、delete (删除)、view (查看)、batch_create (批量新建)、batch_update (批量更新)、batch_view (批量查看) 和batch_delete (批量删除)。
- 对象描述: 对象的描述信息。

3.6.2 Abstract BO

功能介绍

Abstract BO是一个抽象对象, 不能单独存在, 没有数据库表, 需要和业务对象建立继承的关系。建立继承关系后, 业务对象会继承抽象对象中的字段。例如, 抽象对象Abstract和业务对象Role存在继承关系, 在抽象对象Abstract中, 新建一个name字段, 该字段会被Role自动继承。

图 3-102 和业务对象 Role 建立继承关系

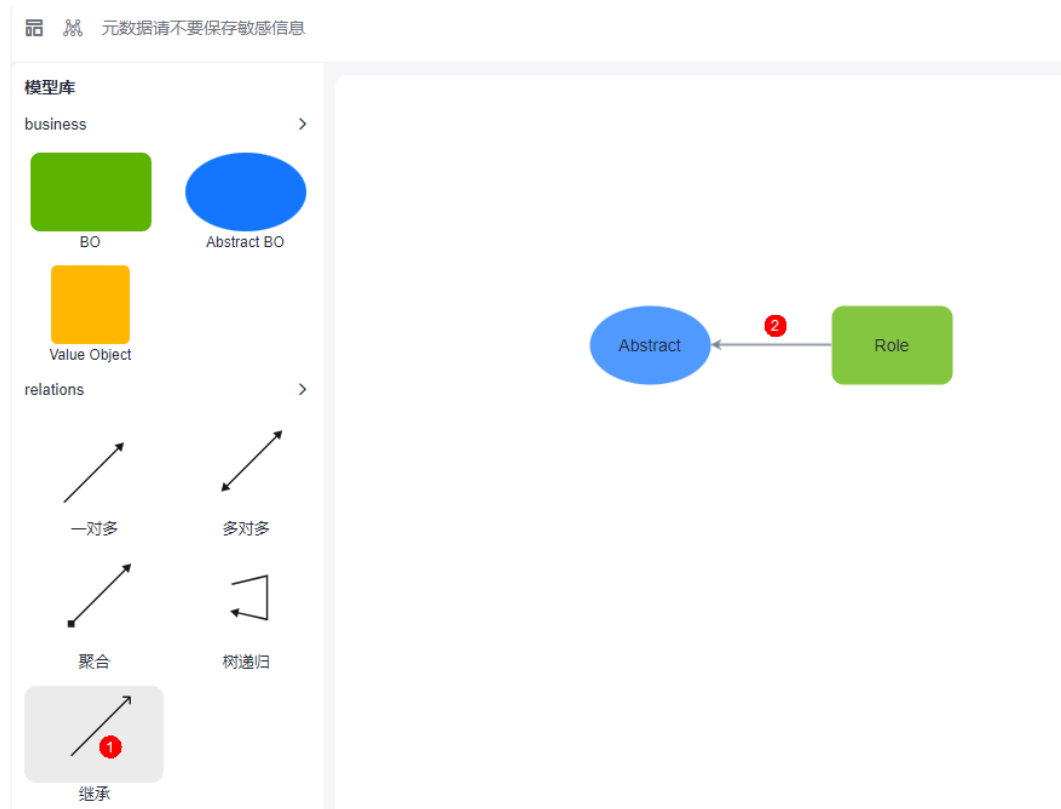


图 3-103 在 Abstract BO 中新建一个 name 字段

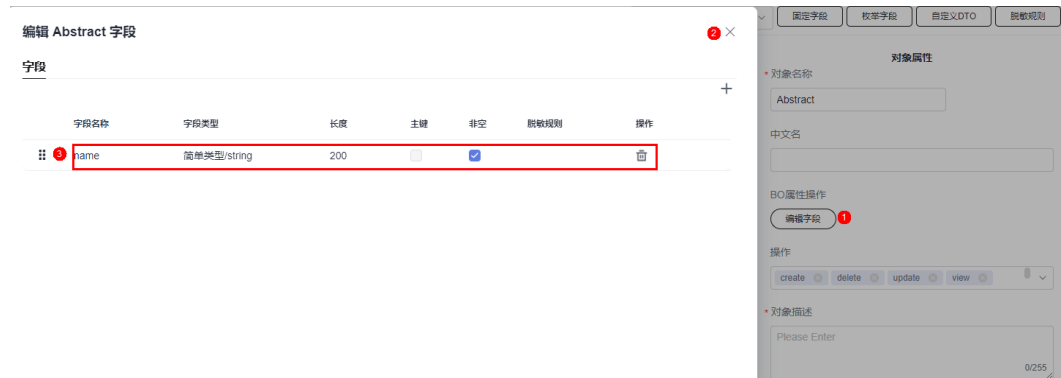


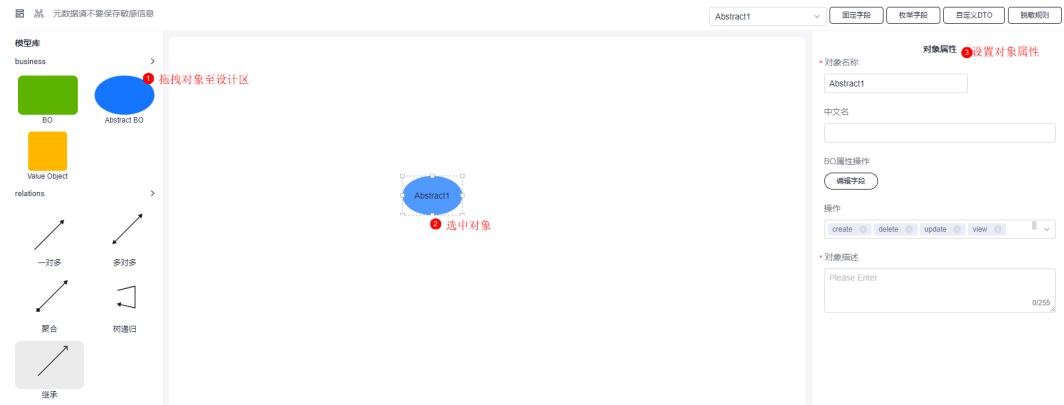
图 3-104 Role 中继承 name 字段



属性说明

在业务设计页面，从“business”中，拖拽“Abstract BO”对象至画布空白区域。选中对象，在右侧页面设置对象属性，如图3-105所示。

图 3-105 Abstract BO



- 对象名称：设置对象的名称，必须使用大驼峰格式，不允许存在连续的大写字母。
- 中文名：设置对象的中文名称。
- BO属性操作：单击“编辑字段”，可以为对象添加所需的字段。
- 操作：对新建的对象可执行哪些操作，如create（新建）、delete（删除）、view（查看）、batch_create（批量新建）、batch_update（批量更新）、batch_view（批量查看）和batch_delete（批量删除）。
- 对象描述：对象的描述信息。

3.6.3 Value Object

功能介绍

Value Object是一个值对象，不能单独存在，需要和业务对象建立聚合的关系。

图 3-106 和业务对象建立聚合关系



聚合后，Value Object中的主键和BO中的主键，共同构成了一个联合的主键。

图 3-107 BO 中的字段



图 3-108 联合主键



属性说明

在业务设计页面，从“business”中，拖拽“Value Object”对象至画布空白区域。选中对象，在右侧页面设置对象属性，如图3-109所示。

图 3-109 Value Object



- 对象名称：设置对象的名称，必须使用大驼峰格式，不允许存在连续的大写字母。
- 中文名：设置对象的中文名称。
- BO属性操作：单击“编辑字段”，可以为对象添加所需的字段。
- 支持BO多租：BO是否支持多租。开启BO多租时，请确保“多租模型”已开启。

说明

购买AstroPro专业版实例时，才会显示“转测BO多租”这个配置项。

图 3-110 开通多租模型



- 不允许修改：对象是否支持修改。
- 操作：对新建的对象可执行哪些操作，如create（新建）、delete（删除）、view（查看）、batch_create（批量新建）、batch_update（批量更新）、batch_view（批量查看）和batch_delete（批量删除）。

- 对象描述：对象的描述信息。

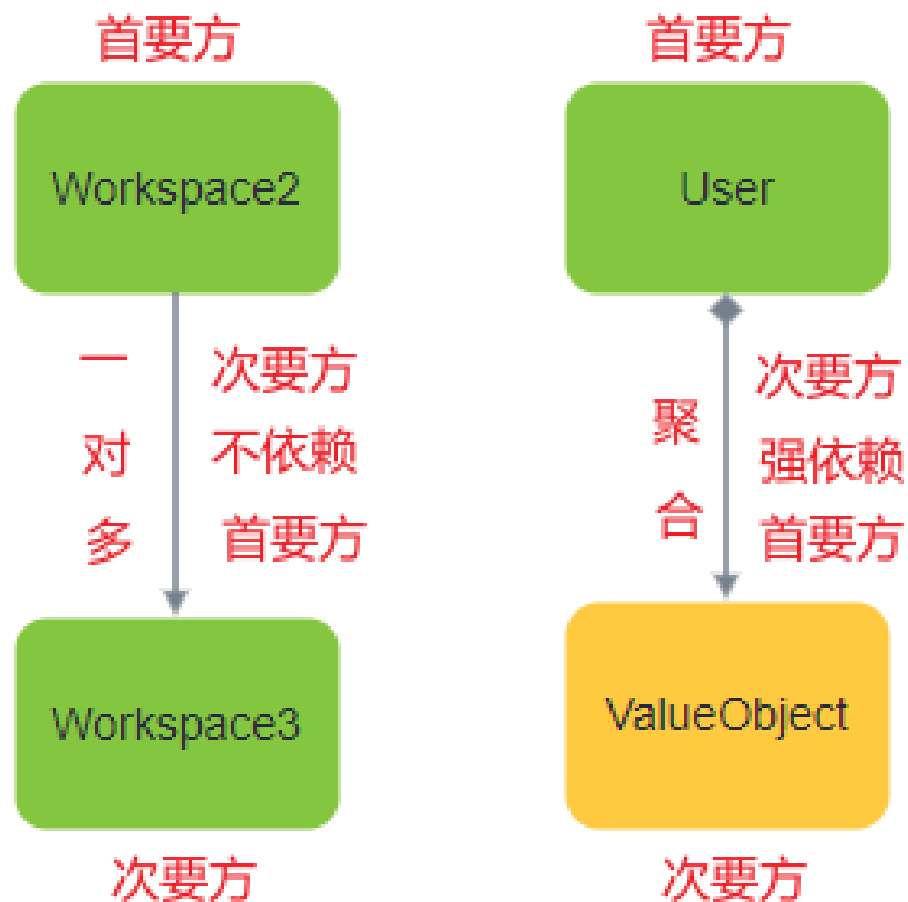
3.6.4 对象间关系

3.6.4.1 一对多

什么是一对多

一对多关系中，次要方可以不依赖于首要方，可以单独存在。删除一对多关系只代表两个对象之间的特定关系的结束，不会影响任何一个对象的生命周期。

图 3-111 一对多和聚合的差异

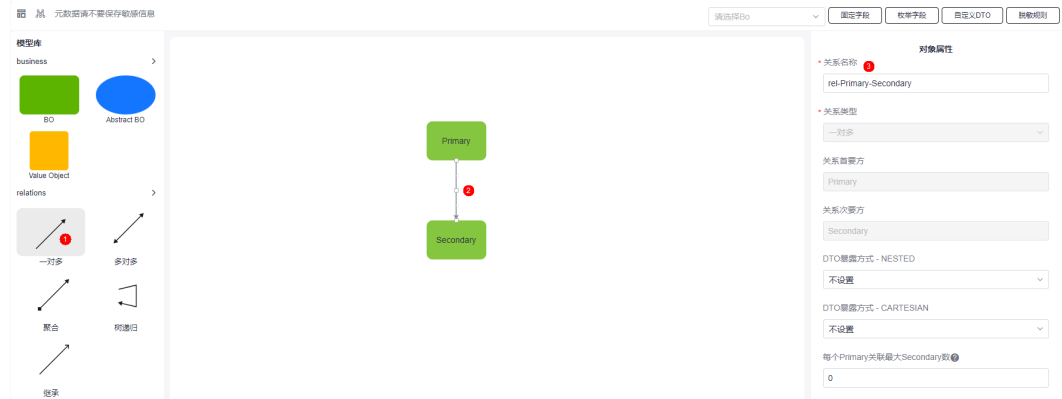


创建一对多关系后，首要方不会做任何的改动，次要方的字段中会自动增加首要方的id，即primaryId，通过这个primaryId去关联首要方的id，从而构建一个一对多的联系。

关系属性设置

在业务设计页面，拖入两个BO业务对象（命名为Primary、Secondary），单击“relations”中的“一对多”，为对象建立一对多关系。选中已创建的关系，在右侧页面即可设置关系属性，如图3-112所示。

图 3-112 一对多关系



- 关系名称：设置一对多关系的名称。
- 关系类型：根据创建的一对多关系自动生成。
- 关系首要方：根据创建的一对多关系自动生成。
- 关系次要方：根据创建的一对多关系自动生成。
- DTO暴露方式 - NESTED：是否设置DTO的NESTED（嵌套）能力。
 - 不设置：不生成NESTED。
 - 只生成DTO：只生成NESTED对象的类。
 - 生成DTO读API：只会生成一个get接口。
 - 生成DTO读写API：除了生成一个get接口，还会生成一个插入接口。
- DTO暴露方式 - CARTESIAN：设置DTO的CARTESIAN（笛卡尔积）能力。
 - 不设置：不生成CARTESIAN。
 - 只生成DTO：只生成CARTESIAN对象的类。
 - 生成DTO读API：只会生成一个get接口。
- 每个Primary关联最大Secondary数：一个首要方和次要方建立关联的数量上限。
- 每个Secondary最大关联Primary数：一个次要方和首要方建立关联的数量上限。
- 每个Primary关联最大Secondary维度上限预警值：首要方一条数据最多关联次要方多少条数据报出告警。

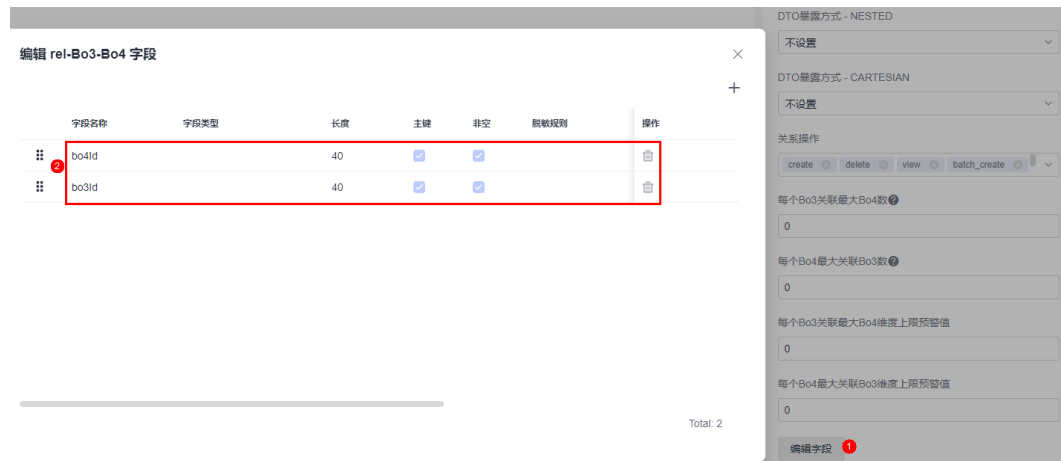
3.6.4.2 多对多

什么是多对多

多对多关系中，首要方和次要方都不会发生任何的变化。唯一的变化是在多对多连线上，会默认添加两个字段用来分别指向关系的两侧。也就是说，一对多和聚合关系是通过在次要方添加主键和外键来表示的，而多对多关系是通过单独创建的关系表来表示的。

和一对多关系一样，删除多对多关系只代表两个对象之间的特定关系的结束，不会影响任何一个对象的生命周期。

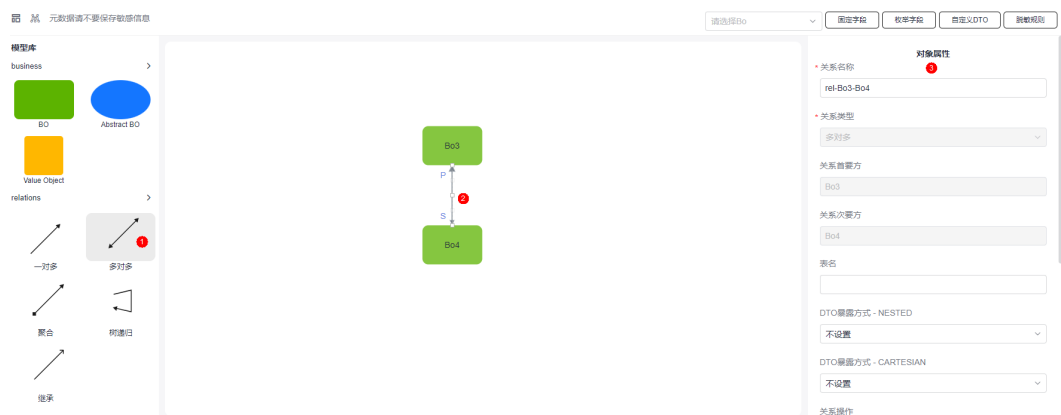
图 3-113 编辑字段



关系属性设置

在业务设计页面，拖入两个BO业务对象（命名为Bo3、Bo4），单击“relations”中的“多对多”，为对象建立多对多关系。选中已创建的关系，在右侧页面即可设置关系属性，如图3-114所示。

图 3-114 多对多关系



- 关系名称：设置多对多关系的名称。
- 关系类型：根据创建的多对多关系自动生成。
- 关系首要方：根据创建的多对多关系自动生成。
- 关系次要方：根据创建的多对多关系自动生成。
- 表名：设置关系表的名称，请按需自定义。在多对多关系中，会建立关系表用来保存首要方和次要方id的关系。
- DTO暴露方式 - NESTED：是否设置DTO的NESTED（嵌套）能力。
 - 不设置：不生成NESTED。
 - 只生成DTO：只生成NESTED对象的类。
 - 生成DTO读API：只会生成一个get接口。
 - 生成DTO读写API：除了生成一个get接口，还会生成一个插入接口。
- DTO暴露方式 - CARTESIAN：设置DTO的CARTESIAN（笛卡尔积）能力。

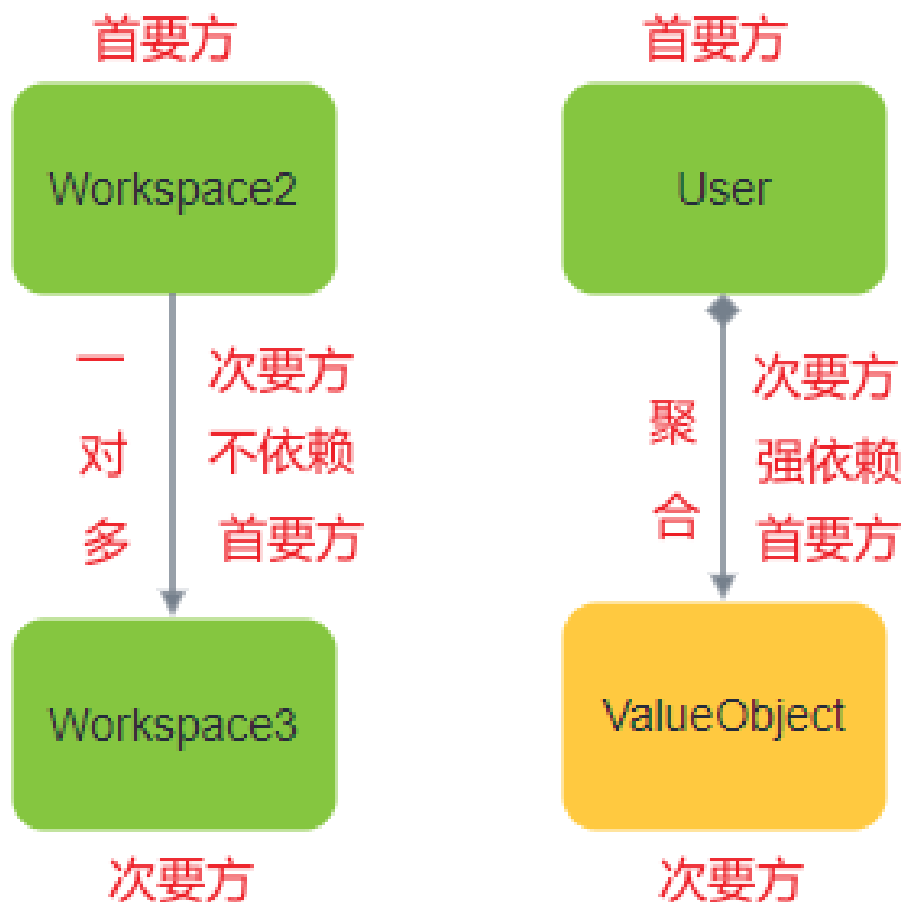
- 不设置：不生成CARTESIAN。
- 只生成DTO：只生成CARTESIAN对象的类。
- 生成DTO读API：只会生成一个get接口。
- 关系操作：对象关系可执行哪些操作，如create（新建）、delete（删除）、view（查看）、batch_create（批量新建）、batch_update（批量更新）、batch_view（批量查看）和batch_delete（批量删除）。
- 每个Bo3关联最大Bo4数：一个首要方和次要方建立关联的数量上限。
- 每个Bo4关联最大Bo3数：一个次要方和首要方建立关联的数量上限。
- 每个Bo3关联最大Bo4维度上限预警值：首要方一条数据最多关联次要方多少条数据报出告警。
- 每个Bo4最大关联Bo3维度上限预警值：次要方一条数据最多关联首要方多少条数据报出告警
- 编辑字段：多对多的关系是通过一个关系表来表示的。单击“编辑字段”，可为关系表添加字段。

3.6.4.3 聚合

什么是聚合

聚合关系本质上也是一种一对多关系，唯一不同的是聚合中次要方必须依赖首要方，任何对于次要方的操作首先要经过首要方才能继续往下操作。

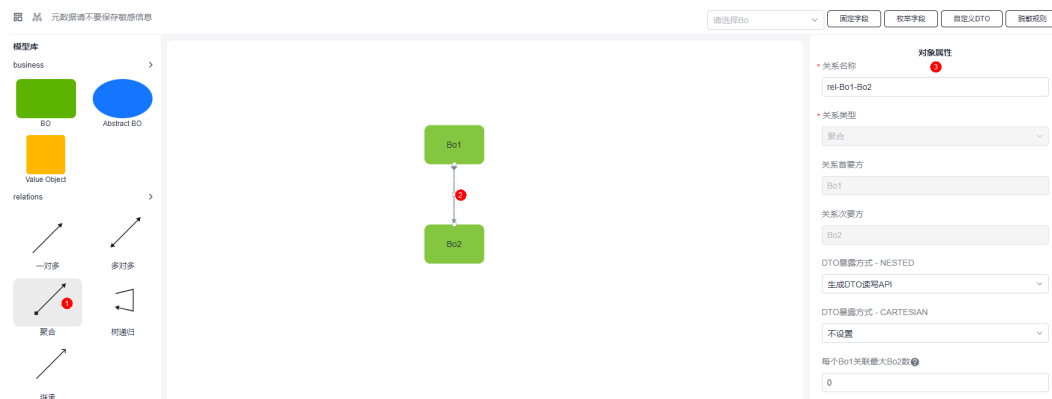
图 3-115 聚合和一对多的差异



关系属性设置

在业务设计页面，拖入两个BO业务对象（命名为Bo1、Bo2）单击“relations”中的“聚合”，为对象建立聚合关系。选中已创建的关系，在右侧页面即可设置关系属性，如图3-116所示。

图 3-116 聚合



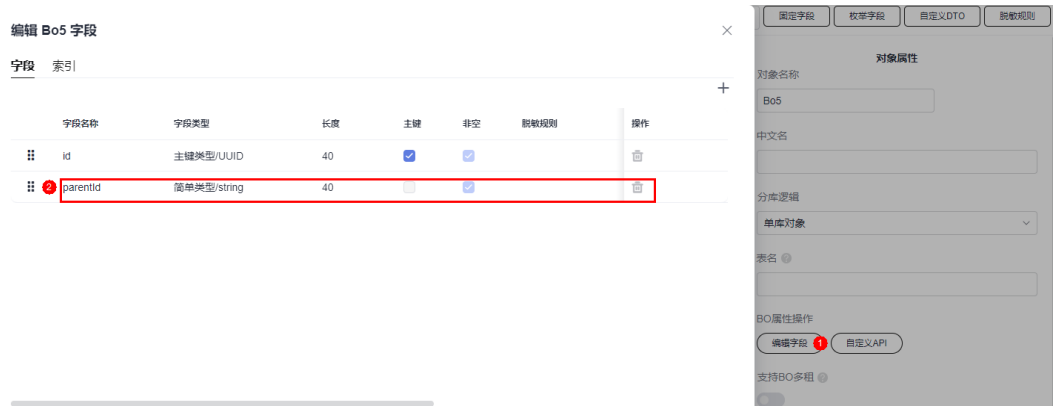
- 关系名称：设置聚合关系的名称。
- 关系类型：根据创建的聚合关系自动生成。
- 关系首要方：根据创建的聚合关系自动生成。
- 关系次要方：根据创建的聚合关系自动生成。
- DTO暴露方式 - NESTED：是否设置DTO的NESTED（嵌套）能力。
 - 不设置：不生成NESTED。
 - 只生成DTO：只生成NESTED对象的类。
 - 生成DTO读API：只会生成一个get接口。
 - 生成DTO读写API：除了生成一个get接口，还会生成一个插入接口。
- DTO暴露方式 - CARTESIAN：设置DTO的CARTESIAN（笛卡尔积）能力。
 - 不设置：不生成CARTESIAN。
 - 只生成DTO：只生成CARTESIAN对象的类。
 - 生成DTO读API：只会生成一个get接口。
- 每个Bo1关联最大Bo2数：一个首要方和次要方建立关联的数量上限。
- 每个Bo1最大关联Bo2数：一个次要方和首要方建立关联的数量上限。
- 每个Bo1关联最大Bo2维度上限预警值：首要方一条数据最多关联次要方多少条数据报出告警。

3.6.4.4 树递归

什么是树递归

树递归和一对多、多对多和聚合之间的差异在于一对多、多对多和聚合是两个不同的业务对象间产生关联，而树递归的双方为相同的对象类型，存储在同一张表中，递归关系的双方可以抽象成父子关系。树递归中，一个对象最多存在一个父对象，类似于数据结构中的树结构。创建树递归后，在数据库中会增加一个parentId字段，用来指向父节点。

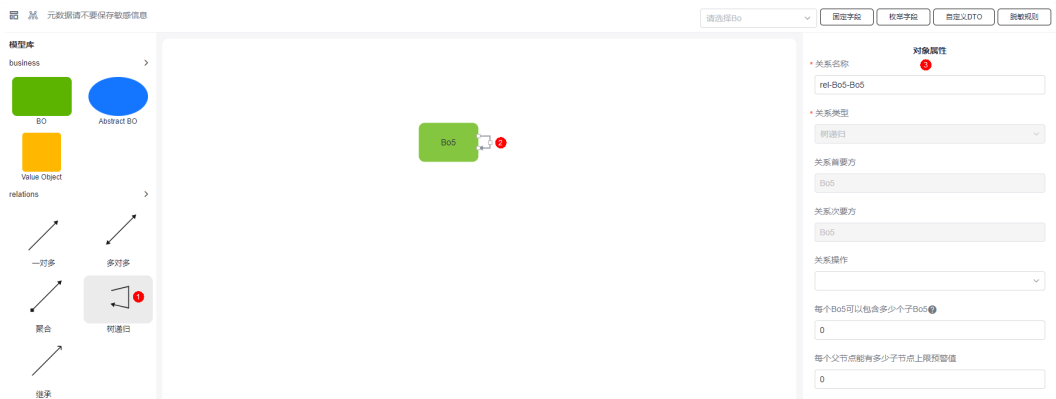
图 3-117 新增 parentId 字段



关系属性设置

在业务设计页面，拖入一个BO业务对象（命名为Bo5）单击“relations”中的“树递归”，为对象建立树递归关系。选中已创建的关系，在右侧页面即可设置关系属性，如图3-118所示。

图 3-118 树递归



- 关系名称：设置树递归关系的名称。
- 关系类型：根据创建的树递归关系自动生成。
- 关系首要方：根据创建的树递归关系自动生成。
- 关系次要方：根据创建的树递归关系自动生成。
- 关系操作：对象关系可执行哪些操作，如create（新建）、delete（删除）、view（查看）、batch_create（批量新建）、batch_update（批量更新）、batch_view（批量查看）和batch_delete（批量删除）。
- 每个Bo5可以包含多少个子Bo5：一个首要方（父节点）和次要方（子节点）建立关联的数量上限。
- 每个父节点能有多少子节点上限预警值：首要方（父节点）一条数据最多关联次要方（子节点）多少条数据报出告警。

3.6.4.5 继承

什么是继承

在继承关系中，业务对象可以继承抽象对象中的所有字段。例如，业务对象Role和抽象对象Abstract，抽象对象中存在name和value两个字段。建立继承关系后，抽象对象Abstract中的字段会被业务对象Role完全继承，如图3-120。

图 3-119 抽象对象 Abstract

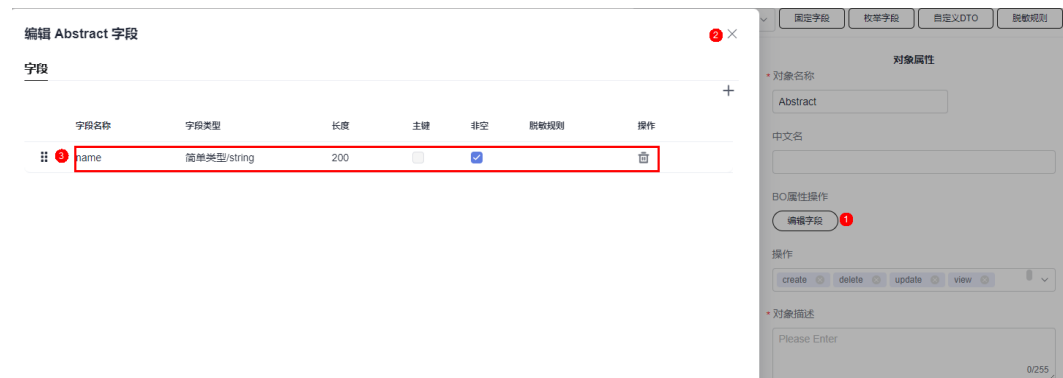


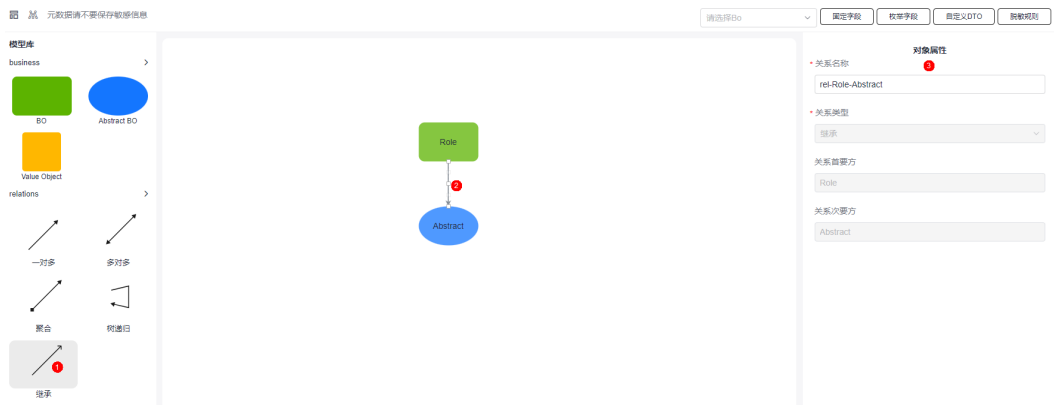
图 3-120 继承 Role 中字段



关系属性设置

在业务设计页面，拖入一个BO业务对象和一个Abstract BO对象（命名为Role、Abstract），单击“relations”中的“继承”，为对象建立继承关系。选中已创建的关系，在右侧页面即可设置关系属性，如图3-121所示。

图 3-121 继承关系



- 关系名称：设置继承关系的名称。
- 关系类型：根据创建的继承关系自动生成。
- 关系首要方：根据创建的继承关系自动生成。
- 关系次要方：根据创建的继承关系自动生成。

4 AstroPro 学堂

4.1 如何自定义 DTO

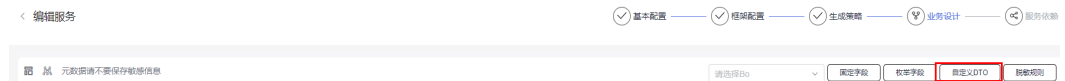
使用说明

自定义DTO相当于数据传输对象，主要用于自定义API时添加参数或返回体。

操作步骤

步骤1 在业务设计页面，单击“自定义DTO”。

图 4-1 自定义 DTO



步骤2 单击 **+**，添加一个自定义DTO。

图 4-2 自定义一个 Dto1



步骤3 在自定义API的参数或返回体中，使用自定义DTO。

1. 从“business”中，拖拽“BO”对象至画布空白区域。
2. 选中BO对象，在对象属性中，单击“自定义API”。

图 4-3 自定义 API



对象属性

- 对象名称:
- 中文名:
- 分库逻辑:
- 表名:
- BO属性操作:
 -
 -
- 支持BO多租

3. 单击 **+**，添加一个自定义API。

图 4-4 自定义一个 API

编辑 Workspace1 自定义API

实例级别	动作名称	请求方法	请求对象	返回对象	路径	描述	操作
类型	action1	get	请点击添加请求对象	请点击添加返回对象	请输入path	--	<input type="button" value="删除"/>

4. 在请求对象或返回对象的参数中，使用自定义DTO。

图 4-5 在自定义 API 中使用自定义 DTO

编辑 Workspace1 自定义API

实例级别	动作名称	请求方法	请求对象	返回对象	路径	描述	操作
类型	action1	get	请点击添加请求对象	请点击添加返回对象	请输入path	--	<input type="button" value="删除"/>

编辑 对象参数

名称	参数类型	参数位置	必选	描述	操作
parameter2	简单类型 / STRING	query	是	--	<input type="button" value="删除"/>

对象类型选择菜单:

- 简单类型
- 对象类型**
- 枚举类型
- 列表类型

业务对象(BO)类型选择菜单:

- 业务对象(BO)类型
- 数据传输对象(DTO)类型**
- 关系对象类型

----结束

4.2 如何定义脱敏规则

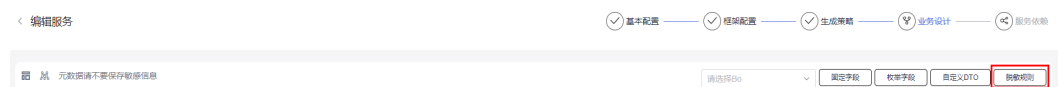
使用说明

当对象字段中存在某些敏感信息时，可通过定义脱敏规则来进行脱敏处理。

操作步骤

步骤1 在业务设计页面，单击“脱敏规则”。

图 4-6 选择脱敏规则



步骤2 单击 **+**，添加一个脱敏规则。

例如，将移动电话的第2位到第5位数字进行脱敏处理。

图 4-7 自定义脱敏规则



步骤3 在对象的字段中，使用脱敏规则。

1. 从“business”中，拖拽“BO”对象至画布空白区域。
2. 选中BO对象，在对象属性中，单击“编辑字段”。
3. 单击 **+**，添加一个phone字段，并将脱敏规则设置为**步骤2**中定义的内容。设置后，返回的值将进行脱敏处理，如1****123456。

图 4-8 引用脱敏规则



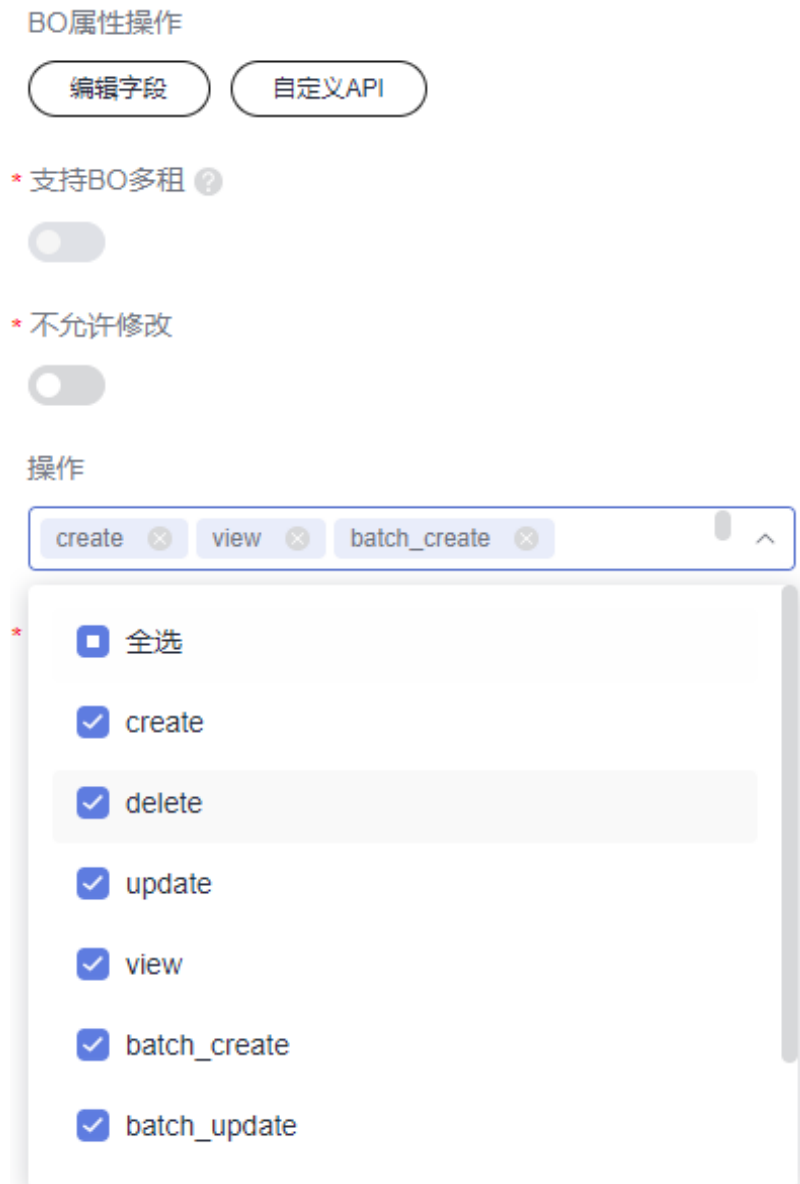
----结束

4.3 如何为对象自定义 API

使用说明

当系统提供的基本操作无法满足需求时，可通过自定义API来实现。

图 4-9 查看操作类型



操作步骤

- 步骤1** 在业务设计页面，选中某个业务对象。
- 步骤2** 单击BO属性操作中的“自定义API”，进入编辑自定义API页面。
- 步骤3** 单击+，按需添加所需的API。

图 4-10 添加 API



- 实例级别：定义API实例的级别，如类型、实例。
- 动作名称：设置API的动作名称。
- 请求方法：HTTP请求方法（也称为操作或动作），用于告诉服务您正在请求什么类型的操作。
 - get：请求服务器返回指定资源。
 - put：请求服务器更新指定资源。
 - post：请求服务器新增资源或执行特殊操作。
 - delete：请求服务器删除指定资源。
- 请求对象：设置请求的对象，即API请求的输入参数。
- 返回对象：请求发送后，您会收到的响应，如状态码。
- 路径：添加API的路径，格式为变量放到“{}”中，单词用“_”连接，非变量单词用“-”连接。
- 描述：自定义API的描述信息，按需进行设置。

步骤4 设置完成后，单击“保存”。

----结束

4.4 如何为对象添加固定字段

使用说明

在AstroPro中，每个业务对象默认只有一个“id”固定字段。除此之外，您还可以为其添加creator、createTime、modifyTime和description四个固定字段。

操作步骤

步骤1 在业务设计页面，选中某个业务对象。

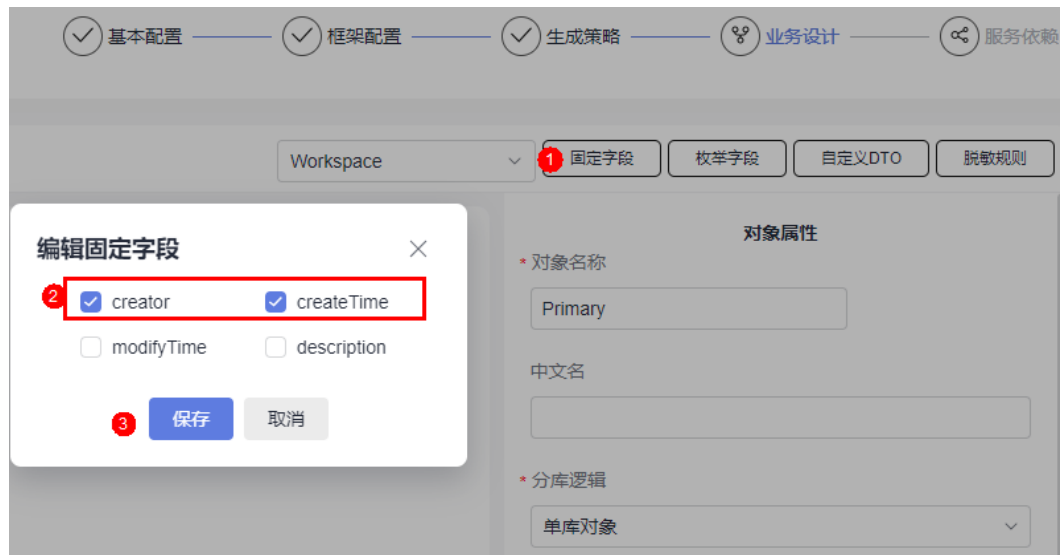
步骤2 单击BO属性操作中的“编辑字段”，查看对象默认字段。
当前仅添加了一个“id”默认字段。

图 4-11 查看默认字段



步骤3 单击“固定字段”，选中待添加的固定字段，单击“保存”。

图 4-12 编辑固定字段



步骤4 选中对象，再次单击BO属性操作中的“编辑字段”，查看对象默认字段。
可查看到对象的默认字段中，除了id还增加了**步骤3**中添加的固定字段。

图 4-13 查看固定字段



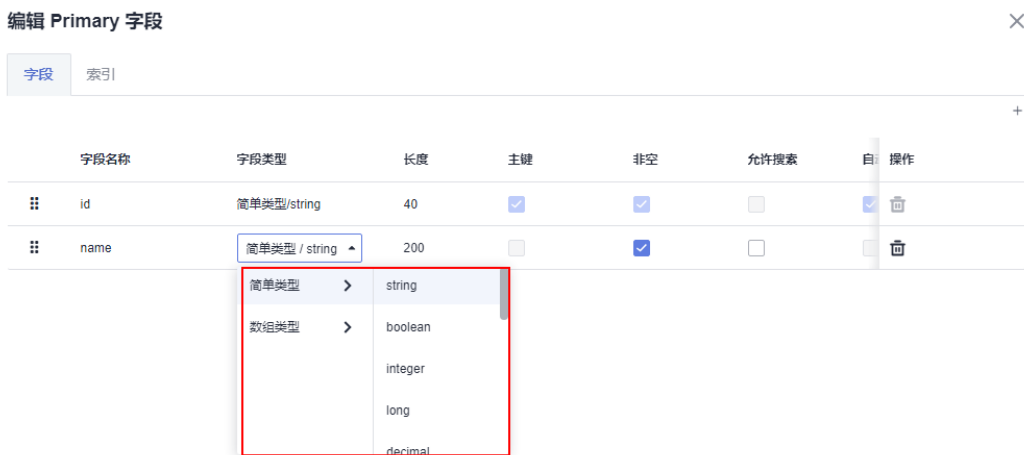
---结束

4.5 如何为对象添加枚举字段

使用说明

AstroPro的对象中仅提供了常用的字段类型（简单类型、数组类型），对于某些特殊的字段类型，如枚举类型（例如性别，男、女），此时可以通过添加枚举字段来实现。

图 4-14 查看字段类型



操作步骤

步骤1 在业务设计页面，单击“枚举字段”。

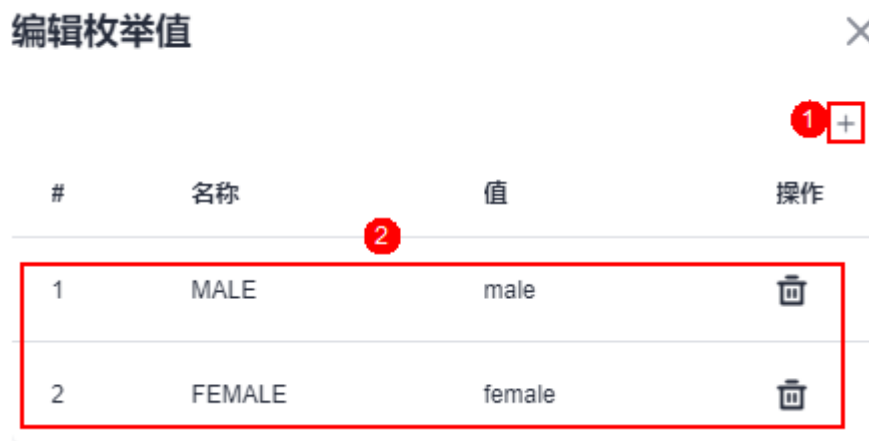
步骤2 在编辑枚举字段页面，单击 **+**，输入枚举名称（如Sex）。

图 4-15 新增枚举字段 Sex



步骤3 单击枚举值下方的“请点击添加枚举值”，添加所需的枚举值。

图 4-16 添加枚举值



步骤4 返回业务设计页面，单击BO属性操作中的“编辑字段”，可查看到字段类型中新增了枚举类型。

图 4-17 字段类型新增了枚举类型



----结束

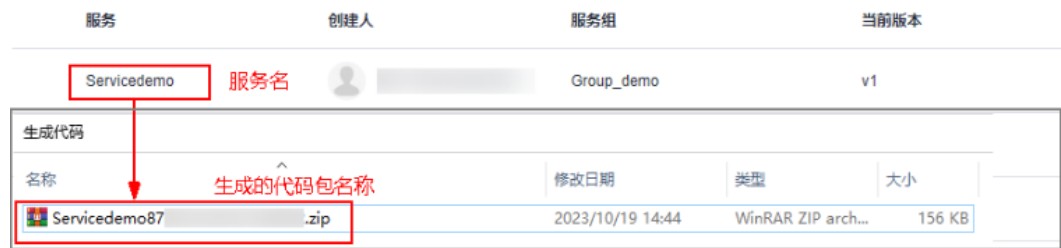
4.6 服务开发框架详解

4.6.1 整体结构介绍

压缩包命名规则

在AstroPro中完成服务的开发后，会生成一个服务代码压缩包，命名为“*服务名称+唯一ID*”。假设服务名称为Servicedemo，则生成的代码压缩包名称，如图4-18所示。

图 4-18 代码压缩包名称



服务	创建人	服务组	当前版本
Servicedemo	服务名	Group_demo	v1

名称	修改日期	类型	大小
Servicedemo87.zip	2023/10/19 14:44	WinRAR ZIP arch...	156 KB

了解代码结构

在AstroPro中编辑服务时，可在“生成策略 > 代码风格”中定义生成代码的工程目录结构，如图4-19、图4-20。

图 4-19 设置代码工程目录结构

代码风格

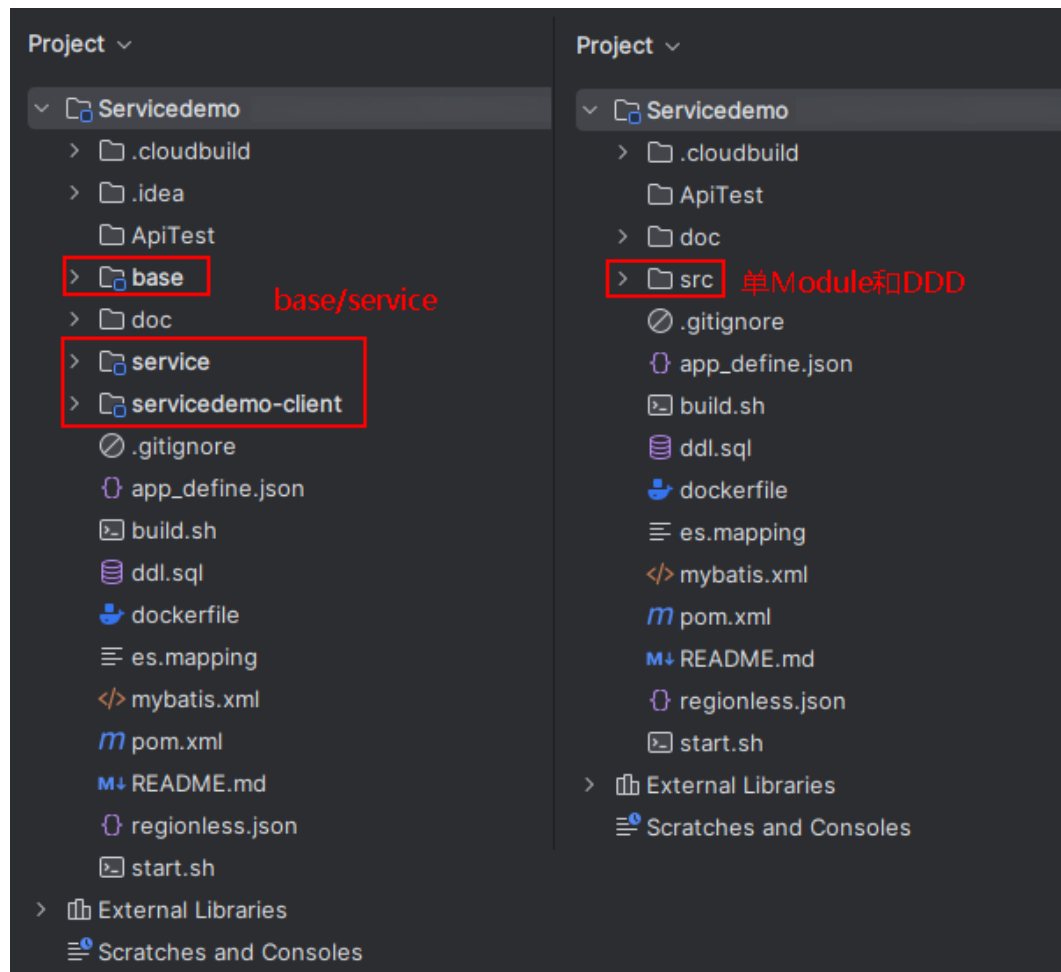
配置文件格式

yaml properties

工程目录

base/service 单Module DDD

图 4-20 工程目录不同类型设置效果



- **单Module、DDD**: 仅会生成“src”一个模块。
- **base/service**: 由base、service和client（如servicedemo-client）三个模块组成。其中，client为客户端模块，在AstroPro中添加服务时，开启“是否生成客户端”配置后（如图4-21），才会生成该模块。

图 4-21 开启“是否生成客户端”

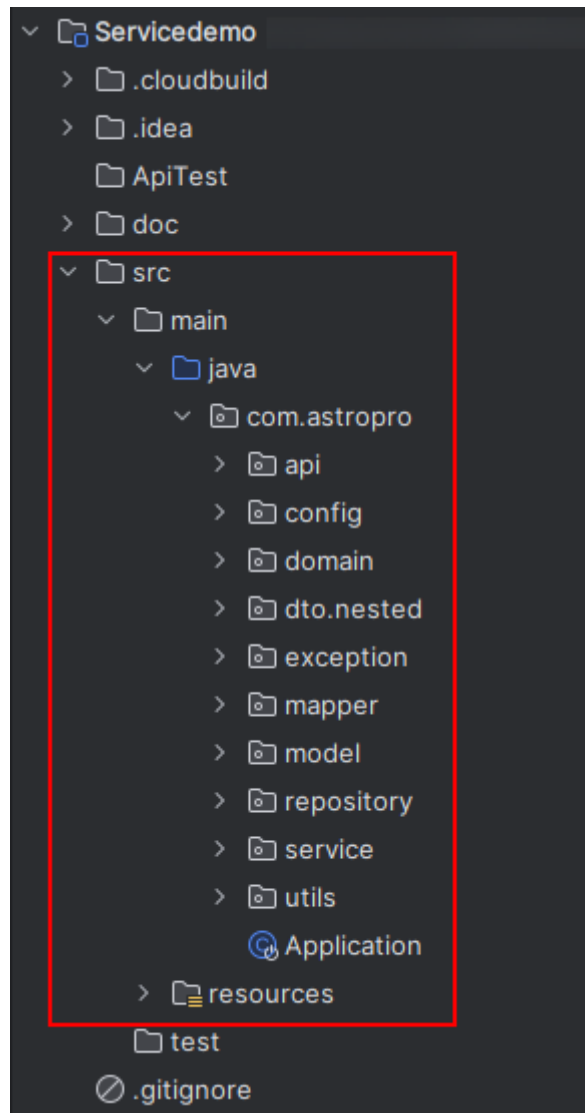


4.6.2 单 Module

工程目录结构

“代码风格 > 工程目录”设置为“单Module”，仅会生成“src”一个模块，如图4-22所示。

图 4-22 单 Module



代码结构说明

代码结构说明中的“{biz}”，为在AstroPro的[业务设计](#)中定义的对象，如BO、Abstract BO等。

com.astropro	
-- api	# API层代码，定义向外部服务暴露的接口（必填项）
{biz}Api.java	
{biz}Controller.java	
-- service	# 承接API直接调用，基本的业务判断逻辑和分发。service层目录，包含接口层和实现层（必须）
-- impl	# service实现代码（必填项）
-- {biz}Service.java	
I{biz}Service.java	# service接口层代码。
-- domain	# 领域层，包含基本的业务和业务聚合（必填项）
{biz}Domain.java	
-- repository	# 数据操作聚合层。包含基类和继承类（必填项）
-- base	# 数据操作聚合层基类代码（必填项）
{biz}BaseRepository.java	
{biz}Repository.java	# 数据操作聚合层继承类代码。用户可在此类中覆写基类中的方法或者增加自定义的方法
-- mapper	# 数据原子操作层。mapper层目录，包含基本接口和继承接口（必填

```

项)
|-- base                                # mapper层基本接口代码（必填项）
    {biz}BaseMapper.java
    {biz}Mapper.java                    # mapper层继承接口代码。用户可在此类中覆写基本接口中的方法或
者增加自定义的方法
|-- model
    |-- entity                          # 实体类（必填项）
    |-- enums                           # 枚举类（可选项）
    |-- qo                               # 查询对象（可选项）
    |-- criteria                         # mybatis查询条件对象（可选项）
|-- dto                                  # 数据传输对象，do组合对象（可选项）
    |-- nested                          # 根据业务对象的关系自动关联生成，嵌套复杂对象（可选项）
    |-- cartesian                       # 根据业务对象的关系自动关联生成，正交的笛卡尔积对象（可选项）
    |-- {customDto}.java                # 用户预先定义好的数据传输对象（可选项）
|-- config                              # 配置类（必填项）
|-- utils                               # 工具类（必填项）
|-- exception                           # 异常类（必填项）
|-- integration                         # 集成第三方服务，隔离外部系统的影响，起防腐作用（可选项）
|-- event                               # 事件层（可选项）
    |-- publish                         # 发布事件的Package，存放事件发布的工具类与 发布的事件对象，屏蔽
技术组件对应应用业务的侵入
    |-- subscribe                       # 订阅事件的Package，存放listener与消费的事件对象，listener只做数
据的监听与数据格式的转换

```

resources 目录结构说明

代码结构说明中的“{biz}”，为在AstroPro的[业务设计](#)中定义的对象，如BO、Abstract BO等。

```

resources
|-- mapper                                # 开源组件mybatis的mapper.xml文件存放目录
    |-- base                            # 该目录下的文件禁止用户改动
        {biz}BaseMapper.xml
        {biz}Mapper.xml                # 只在初次代码生成，用户可在此文件中实现自定义的方法
|-- openapi                              # swagger.yaml存放目录
    swagger.yaml                       # swagger.yaml文件
    application.yml                    # SpringBoot全局配置文件
    banner.txt                         # 应用程序的banner文件
    log4j2.xml                         # log4j2日志配置文件
    metadata.json                      # 元数据配置文件

```

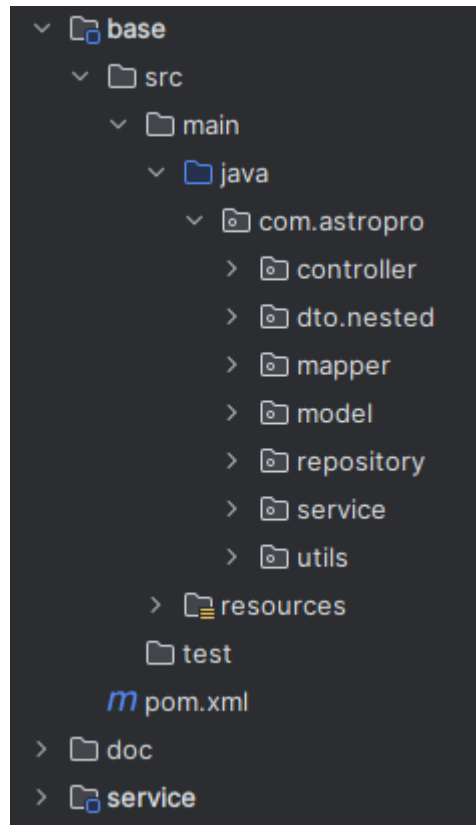
4.6.3 base/service

工程目录结构

“代码风格 > 工程目录”设置为“base/service”时，会生成base、service和client（如servicedemo-client）三个模块。其中，client为客户端模块，在AstroPro中添加服务时，开启“是否生成客户端”配置后才会生成。

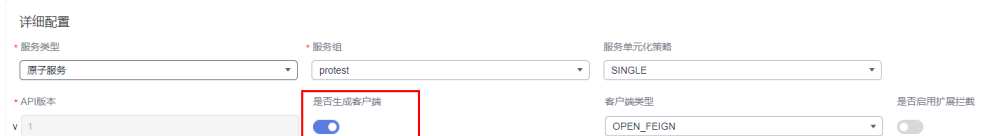
- base: AstroPro自动生成出来的，用户不可以修改。但是可以使用AstroPro预置的devspore-codegen-maven-plugin插件，通过修改本地metadata.json元数据，来重新生成base层。使用该插件重新生成base层时，service层不会重新生成。不建议采用上述方式在本地修改metadata.json，容易引入其他未知错误，建议还是在AstroPro前端重新进行编排生成。

图 4-23 base 层



- service: 当base中提供的功能不足满足需求时，可在service模块进行自定义，即对生成的服务代码进行二次开发。
- client: 客户端模块，在AstroPro中添加服务时，开启“是否生成客户端”配置后（如图4-24），才会生成该模块。

图 4-24 开启“是否生成客户端”



base 代码目录结构

代码结构说明中的“{biz}”，为在AstroPro的[业务设计](#)中定义的对象，如BO、Abstract BO等。

com.astropro	
-- controller	# API层代码，定义向外部服务暴露的接口（必填项）
{biz}Api.java	
{biz}Controller.java	
-- service	# 承接API直接调用，基本的业务判断逻辑和分发。service层目录，包含接口层（必填项）
I{biz}Service.java	# service接口层代码
-- repository	# 数据操作聚合层（必填项）
Abstract{biz}Repository.java	# 数据操作聚合层代码。
-- mapper	# 数据原子操作层。mapper层目录，包含基本接口（必填项）
{biz}Mapper.java	# mapper层接口代码
-- model	# 业务对象层，包含实体类、枚举类、查询对象和mybatis查询条件对象

<code>{biz}.java</code>	# 实体类（可选项）
<code>{xxx}Enum.java</code>	# 枚举类（可选项）
<code>{biz}Qo.java</code>	# 查询对象（可选项）
<code>{biz}Criteria.java</code>	# mybatis查询条件对象（可选项）
<code>-- dto</code>	# 数据传输对象，do组合对象（可选项）
<code> -- nested</code>	# 根据业务对象的关系自动关联生成，嵌套复杂对象（可选项）
<code> -- cartesian</code>	# 根据业务对象的关系自动关联生成，正交的笛卡尔积对象（可选项）
<code> -- {customDto}.java</code>	# 用户预先定义好的数据传输对象（可选项）
<code>-- utils</code>	# 工具类（必填项）
<code>-- resources</code>	# 资源目录结构
<code> -- mapper</code>	# 开源组件mybatis的mapper.xml文件存放目录
<code> {biz}Mapper.xml</code>	# 该目录下的文件禁止用户改动

base 资源目录结构

代码结构说明中的“{biz}”，为在AstroPro的**业务设计**中定义的对象，如BO、Abstract BO等。

<code>resources</code>	# 资源目录结构
<code>-- mapper</code>	# 开源组件mybatis的mapper.xml文件存放目录
<code> {biz}Mapper.xml</code>	# 该目录下的文件禁止用户改动

service 层代码结构

代码结构说明中的“{biz}”，为在AstroPro的**业务设计**中定义的对象，如BO、Abstract BO等。

<code>com.astropro</code>	
<code>-- service</code>	# 承接API直接调用，基本的业务判断逻辑和分发。service层目录，只包含实现层，用户可自定义实现service层逻辑（必填项）
<code> {biz}Service.java</code>	# service实现代码（必填项）
<code>-- repository</code>	# 数据操作聚合层（必填项）
<code> {biz}Repository.java</code>	# 数据操作聚合层继承类代码。用户可在此类中覆写基类中的方法或者增加自定义的方法
<code>-- mapper</code>	# 数据原子操作层，mapper层目录（必填项）
<code> {biz}CustomMapper.java</code>	# mapper层用户自定义mapper接口代码，用户可在此类中用户可在该类中实现自定义mapper接口
<code>-- enums</code>	# 枚举类（必填项）
<code>-- config</code>	# 配置类（必填项）
<code>-- utils</code>	# 工具类（必填项）
<code>-- exception</code>	# 异常类（必填项）
<code>-- integration</code>	# 集成第三方服务，隔离外部系统的影响，起防腐作用（可选项）
<code>-- event</code>	# 事件层（可选项）
<code> -- publish</code>	# 发布事件的Package，存放事件发布的工具类与发布的事件对象，屏蔽技术组件对应用业务的侵入
<code> -- subscribe</code>	# 订阅事件的Package，存放listener与消费的事件对象，listener只做数据的监听与数据格式的转换

service 层资源目录结构

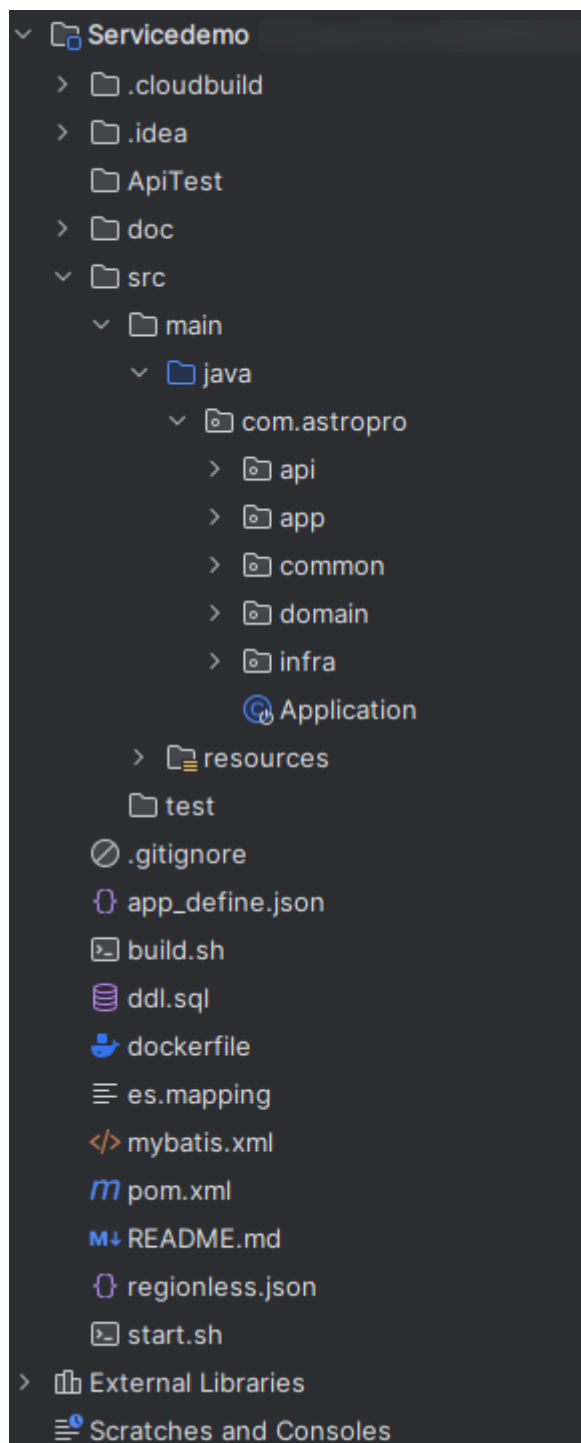
代码结构说明中的“{biz}”，为在AstroPro的**业务设计**中定义的对象，如BO、Abstract BO等。

<code>resources</code>	# 资源目录结构
<code>-- mapper</code>	# 开源组件mybatis的mapper.xml文件存放目录
<code> {biz}CustomMapper.xml</code>	# 用户可在此文件中实现自定义的方法
<code>-- openapi</code>	# swagger.yaml存放目录
<code> swagger.yaml</code>	# swagger.yaml文件
<code> application.yml</code>	# SpringBoot全局配置文件
<code> banner.txt</code>	# 应用程序的banner文件
<code> log4j2.xml</code>	# log4j2日志配置文件
<code> metadata.json</code>	# 元数据配置文件

4.6.4 DDD

“代码风格 > 工程目录” 设置为 “DDD” 时，和 “单Module” 一样，仅会生成 “src” 一个模块，如[图4-25](#)所示。

图 4-25 DDD



代码结构说明

代码结构说明中的“{biz}”，为在AstroPro的**业务设计**中定义的对象，如BO、Abstract BO等。

```

com.astropro
|-- api                                     # API层代码，定义向外部服务暴露的接口（必填项）
    {biz}Api.java                          # swagger注解
    {biz}Controller.java                  # 输入参数非业务校验
|-- app                                     # 应用层（必填项）
    |-- service                            # 应用服务层（必填项）
        I{biz}AppService.java             # 应用服务接口定义（禁止改动）
        |-- impl
            {biz}Service.java             # 返回值转换为dto
    |-- dto                                 # 数据传输对象，用于接口层返回（必填项）
        {biz}Dto.java                     # 与BO一一对应
        {customDto}.java                 # 自定义dto（禁止改动）
        |-- converter                       # bean转换器，用于BO和DTO转换
            {biz}Converter.java           # 与BO是一一对应关系
    |-- consumer                            # 消费者（可选项）
        I{biz}Consumer.java
        {biz2}Handler.java
    |-- producer                            # 生产者（可选项）
        I{biz}Producer.java
    |-- scheduler                           # 定时任务（可选项）
|-- domain                                  # 领域层，包含基本的业务和业务聚合（必填项）
    |-- service
        I{biz}RepositoryService.java      # 领域服务接口定义（devspore生成的持久化服务,禁止修改）
        I{biz}GatewayService.java        # 网关服务接口
        |-- impl
            {biz}DomainService.java       # 自定义的领域服务实现
    |-- entity                              # 该聚合下所有的实体、值对象
        {biz}.java
    |-- model                                # qo、nested、cartesian
    |-- enums                               # 枚举类
    |-- event                               # 领域事件实体定义
        {xxx}Event.java
|-- infra                                   # 基础设施层
    |-- repository                          # 数据操作聚合层，包含基类和继承类（必填项）
        |-- base                           # 数据操作聚合层基类代码（必填项）
            {biz}BaseRepository.java
            {biz}Repository.java          # 数据操作聚合层继承类代码，并实现对应的domainService
        |-- mapper                          # 数据原子操作层。mapper层目录，包含基本接口和继承接口（必填项）
            |-- criteria                    # mybatis查询对象
            |-- base                        # mapper层基本接口代码（必填项）
                {biz}BaseMapper.java
                {biz}Mapper.java         # mapper层继承接口代码。用户可在此类中覆写基本接口中的方法
            或者增加自定义的方法
            |-- integration                 # 防腐层，集成第三方服务（跨进程的外部服务），隔离外部系统的影响（可选项）
                GatewayService.java      # 与领域服务层的I{biz}GatewayService.java一一对应
            |-- mq                           # mq的具体实现，如kafka、rocketmq
|-- common                                  # 通用包
    |-- config                              # 配置类（必填项）
    |-- utils                               # 工具类（必填项）
    |-- exception                          # 异常类（必填项）
    |-- filter                              # 过滤器（可选项）
    |-- interceptor

```

资源目录说明

代码结构说明中的“{biz}”，为在AstroPro的**业务设计**中定义的对象，如BO、Abstract BO等。

```

resources
|-- mapper                                # 开源组件mybatis的mapper.xml文件存放目录
|-- base                                  # 该目录下的文件禁止用户改动

```

```

    {biz}BaseMapper.xml
    {biz}Mapper.xml
-- openapi
    swagger.yaml
    application.yml
    banner.txt
    log4j2.xml
    metadata.json
    # 只在初次代码生成，用户可在此文件中实现自定义的方法
    # swagger.yaml存放目录
    # swagger.yaml文件
    # SpringBoot全局配置文件
    # 应用程序的banner文件
    # log4j2日志配置文件
    # 元数据配置文件

```

4.7 AstroPro-SDK 版本变更与下载

AstroPro-SDK版本变更记录及下载地址如下。您可以按需下载所需的AstroPro-SDK，也可以直接通过[表4-1](#)中链接，一次性下载所有的AstroPro-SDK。

如果您能访问外网并下线maven依赖，则可通过修改maven settings.xml文件来配置华为开源镜像仓库进行依赖管理，具体配置如下：

步骤1 在profiles节点中添加如下内容：

```

<profile>
  <id>MyProfile</id>
  <repositories>
    <repository>
      <id>HuaweiCloudSDK</id>
      <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <id>HuaweiCloudSDK</id>
      <url>https://repo.huaweicloud.com/repository/maven/huaweicloudsdk</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>>false</enabled>
      </snapshots>
    </pluginRepository>
  </pluginRepositories>
</profile>

```

步骤2 在mirrors节点中增加：

```

<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*,!HuaweiCloudSDK</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven</url>
</mirror>

```

步骤3 增加activeProfiles标签激活配置：

```

<activeProfiles>
  <activeProfile>MyProfile</activeProfile>
</activeProfiles>

```

----结束

表 4-1 SDK 全量下载

开发语言	包含的模块	版本及下载地址	SHA256值	变更描述
Java	所有	obs-20240525	e6aec815e76ee0b12624a5238234309e7173210a528949f2b65503802c51981d	2024年05月25日发布版本。
Java	所有	obs-20240330	b5724214b6b56645d2517fa21bfec8eb8328b762999ee4bc1ee358b060d109e1	2024年03月30日发布版本。

表 4-2 devspore-auth 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	devspore-auth-oneaccess	3.10.9.JDK17-RELEASE 3.10.8.JDK8-RELEASE	修复一些问题。
Java		3.10.7.JDK17-RELEASE 3.10.7.JDK8-RELEASE	修复一些问题。
Java		3.10.6.JDK17-RELEASE 3.10.6.JDK8-RELEASE	修复一些问题。
Java		3.10.5.JDK17 3.10.5.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java		3.10.3.OBT.JDK8-RELEASE	初次发布。

表 4-3 devspore-security 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	<ul style="list-style-type: none"> devspore-security devspore-security-commons devspore-security-parent 	2.1.9.JDK17-RELEASE 2.1.5.JDK8-RELEASE	修复一些问题。
Java		2.1.4.JDK17-RELEASE 2.1.4.JDK8-RELEASE	修复一些问题。
Java		2.1.3.JDK17-RELEASE 2.1.3.JDK8-RELEASE	修复一些问题。
Java		2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java		2.0.3.OBT.JDK8-RELEASE	初次发布。

表 4-4 devspore-probe 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	devspore-probe	2.1.8.JDK17-RELEASE 2.1.7.JDK8-RELEASE	修复一些问题。
Java		2.1.4.JDK17-RELEASE 2.1.4.JDK8-RELEASE	修复一些问题。
Java		2.1.3.JDK17-RELEASE 2.1.3.JDK8-RELEASE	修复一些问题。
Java		2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java		2.0.2.JDK8-RELEASE	修复一些问题。
Java		2.0.1.JDK8-RELEASE	初次发布。

表 4-5 devspore-clientcontrol 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	devspore-clientcontrol	2.1.4.JDK17-RELEASE 2.1.4.JDK8-RELEASE	修复一些问题。
Java		2.1.3.JDK17-RELEASE 2.1.3.JDK8-RELEASE	<ul style="list-style-type: none"> • 新增JDK17。 • 修复一些问题。
Java		2.0.3.JDK8-RELEASE	初次发布。

表 4-6 devspore-http-log 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	devspore-http-log	2.1.4.JDK17-RELEASE	修复一些问题。此次只更新了JDK17。
Java		2.1.3.JDK17-RELEASE 2.1.3.JDK8-RELEASE	修复一些问题。
Java		2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> • 新增JDK17。 • 修复一些问题。
Java		2.0.3.JDK8-RELEASE	修复一些问题。
Java		2.0.1.JDK8-RELEASE	初次发布。

表 4-7 devspore-mas-common 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	devspore-mas-common	2.1.3.JDK17-RELEASE 2.1.3.JDK8-RELEASE	修复一些问题。
Java		2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> • 新增JDK17。 • 修复一些问题。
Java		2.0.1.JDK8-RELEASE	初次发布。

表 4-8 devspore-dds 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	<ul style="list-style-type: none"> devspore-dds devspore-dds-parent 	2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java	<ul style="list-style-type: none"> spring-cloud-starter-huawei-devspore-dds 	2.0.2.JDK8-RELEASE	初次发布。

表 4-9 devspore-css 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	devspore-css	2.0.1.JDK8-RELEASE	初次发布。

表 4-10 devspore-datasource 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	<ul style="list-style-type: none"> devspore-datasource 	2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java	<ul style="list-style-type: none"> devspore-datasource-parent spring-cloud-starter-huawei-devspore-datasource 	2.0.2.JDK8-RELEASE	初次发布。

表 4-11 devspore-dcs 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	<ul style="list-style-type: none"> devspore-dcs devspore-dcs-parent 	2.1.3.JDK17-RELEASE 2.1.3.JDK8-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> spring-cloud-starter-huawei-devspore-dcs 	2.1.2.JDK17-RELEASE 2.1.2.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java	devspore-dcs	2.0.2.JDK8-RELEASE	初次发布。

表 4-12 spring-boot-huawei 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	<ul style="list-style-type: none"> spring-boot-huawei 	2.1.14.JDK17-RELEASE 2.1.10.JDK8-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> spring-boot-huawei-dependencies 	2.1.8.JDK17-RELEASE 2.1.8.JDK8-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> spring-boot-huawei-parent 	2.1.6.JDK17-RELEASE 2.1.6.JDK8-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> spring-boot-starter-huawei 	2.1.4.JDK17-RELEASE 2.1.4.JDK8-RELEASE	<ul style="list-style-type: none"> 新增JDK17。 修复一些问题。
Java		2.0.3.JDK8-RELEASE	初次发布。

表 4-13 devspore-generator 版本变更

开发语言	包含的模块	版本及下载地址	变更描述
Java	<ul style="list-style-type: none"> devspore-core 	2.1.15-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> devspore-generator 	2.1.11-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> devspore-horizon 	2.1.8-RELEASE	修复一些问题。
Java	<ul style="list-style-type: none"> devspore-metadata devspore-parent devspore-plugin devspore-util 	2.0.2.JDK8-RELEASE	初次发布。

SDK 压缩包完整性校验

- linux下验证

步骤1 在表格的“版本及下载地址”中获取SDK包下载路径。

步骤2 下载SDK包到本地。

步骤3 输入如下命令。

```
sha256sum {压缩包名}
```

步骤4 对比压缩包.sha256的SHA256值和下载后的SDK包的SHA256值。

- 一致，则表示压缩包完整，下载过程不存在篡改和丢包。

- 不一致，说明SDK压缩包被篡改，需要重新获取。

----结束

- windows验证

步骤1 在表格的“版本及下载地址”中获取SDK包下载路径。

步骤2 下载SDK包到本地。

步骤3 打开本地命令提示符框，输入如下命令。

```
certutil -hashfile {压缩包名} SHA256
```

步骤4 对比压缩包.sha256的SHA256值和下载后的SDK包的SHA256值。

- 一致，则表示压缩包完整，下载过程不存在篡改和丢包。
- 不一致，说明SDK压缩包被篡改，需要重新获取。

----结束