CodeArts API 用户指南

CodeArts API 用户指南

文档版本 01

发布日期 2024-04-10

华为技术有限公司

商标声明


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

注意

您购买的产品、服务或特性等应受华为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，华为公司对本文档内容不做任何明示或暗示的声明或保证。

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

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： https://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 全局代理

CodeArts API客户端版本支持设置全局网络代理，设置后，代理应用于所有联网服务，包括连接CodeArts API服务器和发送接口请求。

步骤1 打开CodeArts API客户端后，单击右上角 ，进入设置页面。

步骤2 单击“网络代理”，进入网络代理设置页面，可根据需要选择代理模式，包括：使用系统代理（默认）、不使用代理、自定义。

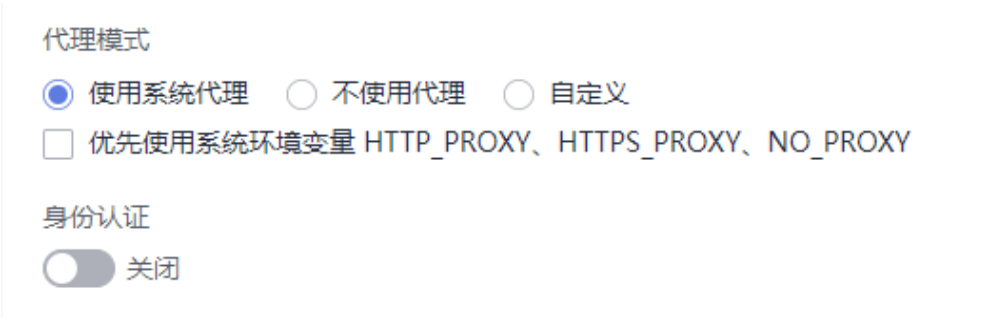
----结束

全局代理设置

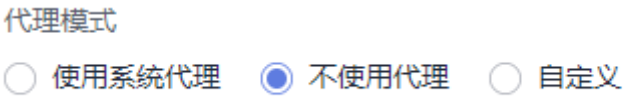
- 使用系统代理

  进入网络代理设置页面后，默认使用系统代理。

  - 可根据需要选择是否优先使用系统环境变量HTTP_PROXY、HTTPS_PROXY、NO_PROXY，勾选后，CodeArts API会优先调用系统环境变量。
  - 身份认证：请查看**身份认证**。



- 不使用代理
  选择不使用代理时，此时CodeArts API不使用任何代理（包含系统代理）连接服务器。

- 自定义

  选择自定义代理时，需配置“接口类型”、“代理服务器”、“身份认证”、“Proxy Bypass”。
  - 接口类型：接口类型默认HTTP和HTTPS全选，支持两种协议类型的代理。
  - 代理服务器：需输入代理服务器地址和端口。
  - 身份认证：请查看**身份认证**。
  - Proxy Bypass：无需进行代理转发的IP或域名，可填入多个，以英文逗号进行分隔。

身份认证

在选择“使用系统代理”与“自定义”代理两种代理模式时，支持配置身份认证信息。

开启配置身份认证开关后，CodeArts API需要重启客户端使身份认证信息配置生效。

输入用户名和密码（密码支持明文显示），然后单击“保存并重启CodeArts API”即可。

2 API 设计

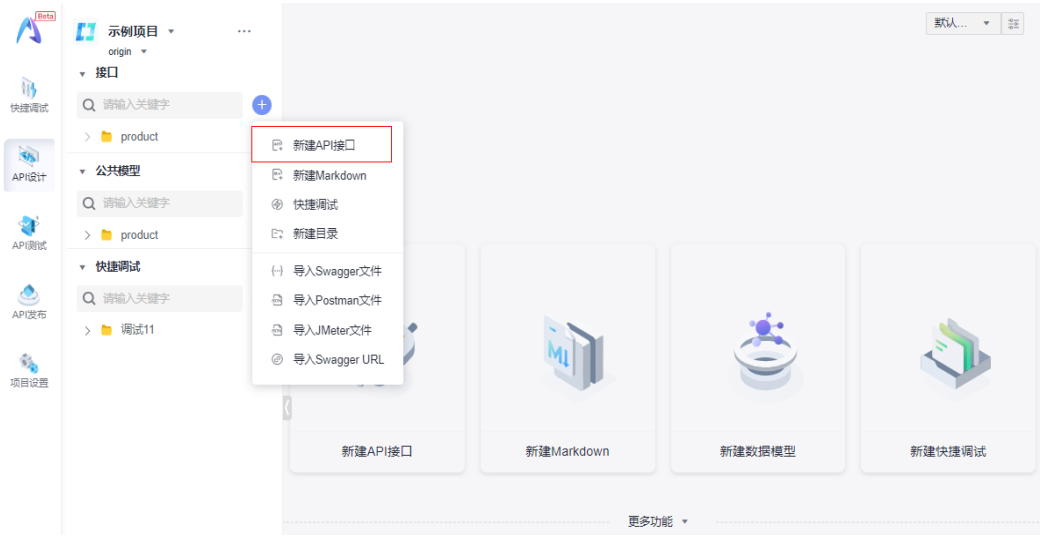
2.1 API 设计

API（应用程序编程接口）设计是API全生命周期的第一步，其含义是通过规定接口的路径、所属目录、Tag、请求参数、请求体等来定义接口，从而帮助开发人员在沟通中对某一接口有着同样、确定的定义。

新建 API

进入CodeArts API目标项目后，单击主页面的“新建API接口”选项，创建http类型接口。

也可以单击左侧搜索框旁边的 ，选择下拉选项中的“新建API接口”，创建http类型接口。



设计 API

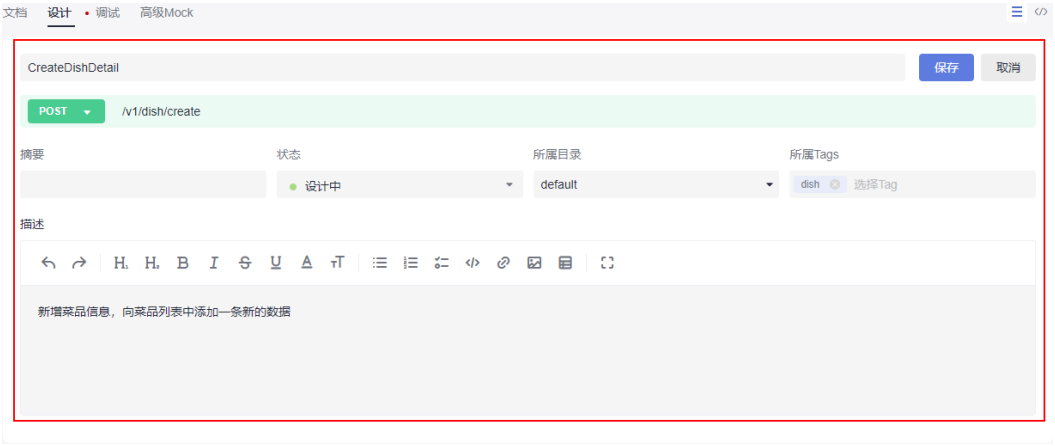
接口文档应针对以下要素进行设计：

- 接口基本信息
- 接口路径
- 请求方式
- 接口请求参数
- 接口返回响应
- 安全方案

接口基本信息

填写接口的基本信息，包括：

- 名称：接口名称。
- 摘要：接口的摘要信息。
- 状态：API生命周期的状态，包括设计中、联调中、测试中、测试完、已发布、将废弃、已废弃。
- 所属目录：接口在项目中所属的目录。
- 所属Tags：接口所属的Tags，可以直接选择所属目录信息当作所属Tags，也可以在所属Tags框中手动输入Tag名再回车来生成所属Tags。
- 描述：可添加接口的相关描述，包括接口功能、使用注意事项、使用场景定义等详细描述信息，接口描述支持通用Markdown语言编辑。



接口路径

接口的URL，客户端可以通过接口路径向服务端发起请求。以“/”开头，如：“/car”、“/car/{owner}”。

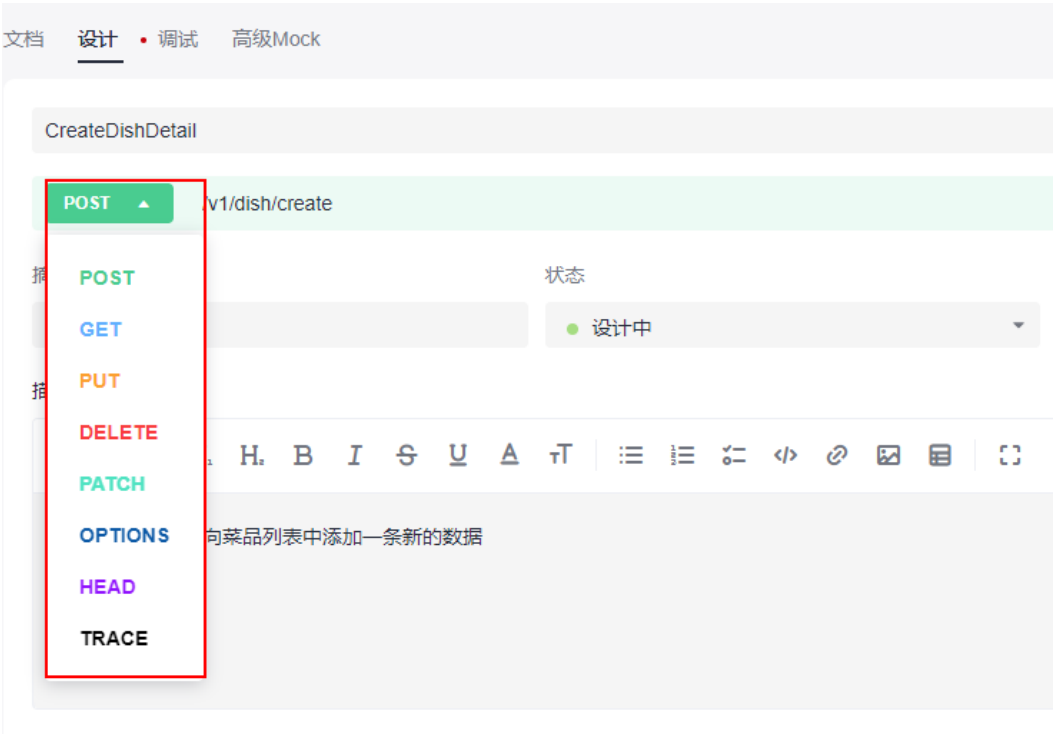
说明

- 接口路径一般不包含域名和http、https协议名，如需加上域名进行调试，在调试功能页面有相应处理。
- 大括号“{}”中间的字符串表示Path参数。
- 不支持通过接口路径来添加Query参数，如：“/car?owner=zhangsan”。

请求方式

定义接口的请求方式，制定了客户端对服务端资源操作的方法，在API设计过程中，需要根据业务和操作类型选择合适的请求方式。

- GET（获取）：用于获取指定资源，使用查询参数传递参数，适用于读取资源和查询数据。
- POST（提交）：用于提交数据。POST请求可以传递请求体参数，适用创建新资源、提交表单数据或执行某些操作等场景。
- PUT（更新）：用于更新或替换服务端的指定资源。
- DELETE（删除）：用于删除服务端的指定资源。
- OPTIONS（选项）：用于获取服务器支持的http方法和资源的相关信息。可用于客户端与服务端之间的握手过程，了解服务器所支持的方法和功能。
- HEAD（请求头）：与GET类似，但只返回响应头部，不返回实体内容，用于获取资源的元信息，如：文件大小、修改日期等。
- PATCH（补丁）：用于对资源进行局部更新。PATCH请求类似于PUT请求，但是只更新资源的一部分。
- TRACE（跟踪）：用于简历与代理服务器的隧道连接，通常用于进行安全的SSL/TLS加密通信。



接口请求参数

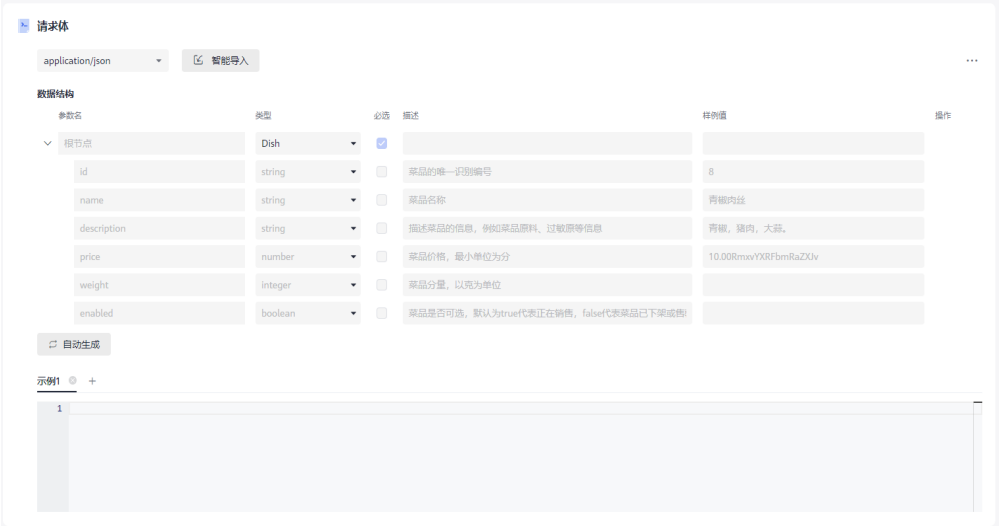
- Params、Path、Header、Cookies
  - Query参数：接口请求中的一种参数传递方式，它通常用于传递一些可选的参数，比如过滤条件、排序方式、分页参数等。在URL中表现为末尾“?”后的字符串（如：“/car?owner=zhangsan”，那么“owner=zhangsan”即Query参数，其中“owner”为参数的key，“zhangsan”为参数的value）。
  - Path参数：也称为“路径参数”，是API请求中的一种参数传递方式。在URL中表现为大括号“{}”括起来的字符串（如：“/car/{owner}”，那么“{owner}”表示key为“owner”的Path参数）。
  - Header参数：请求头中的参数。
  - Cookies：类型为“小型文本文件”，是某些网站为了辨别用户身份，进行Session跟踪而储存在用户本地终端上的数据（通常经过加密），由用户客户端计算机暂时或永久保存的信息。
  - 前置脚本：CodeArts API支持接口前置脚本操作，详见**2.7 脚本能力**。
  - 后置脚本：CodeArts API支持接口后置脚本操作，详见**2.7 脚本能力**。

- 请求体

  在发起http请求时，需要带上一些参数以便服务器处理，通过http请求体传递的参数被称为Body参数。Body参数包含在请求的主体部分中，通常是一些表单数据、

JSON数据或者二进制数据。在发送请求时，会根据Body参数类型，自动在请求Header中添加对应Content-Type参数。若手工设置Content-Type的类型，则必须与Body的参数类型匹配，否则系统会自动忽略掉手动设置的Content-Type值。

- none：无body参数。
- application/json：json格式数据。Content-Type为 “application/json”。
- application/xml：xml类型数据，用于标识文件和图像等没提类型，也可以标识结构化数据。Content-Type为 “application/xml”。
- multipart/form-data：表单数据。Content-Type为 “multipart/form-data”。
- application/x-www-form-urlencoded：将表单数据编码后传输到服务器。数据被编码为一系列键值对，每个键值对之间以&连接，并且键与值之间以=分隔。Content-Type为“application/x-www-form-urlencoded” 。



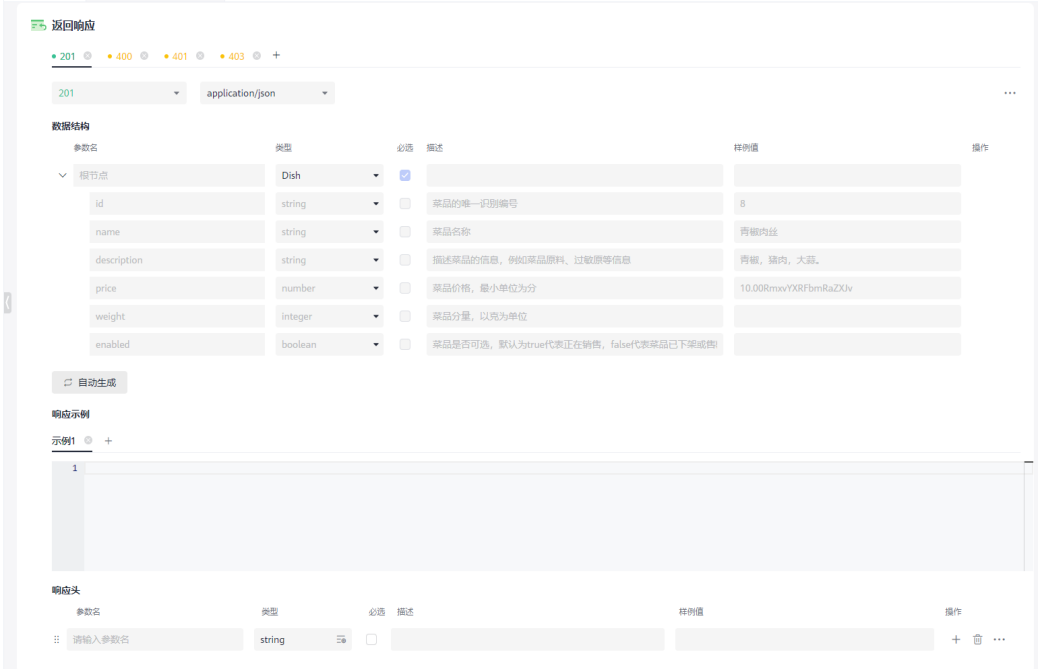
此外，还支持根据Body体数据结构自动生成示例，或将手工编写Body示例智能导入数据结构中。

接口返回响应

返回响应的定义包含：返回响应的状态码、响应内容的数据结构、响应示例和响应头。

- 返回响应的状态码：通过加号来添加运行接口后可能的响应状态码，单击响应状态码可以对状态码进行修改。
- 响应内容的数据结构：规定响应内容的格式，分为“application/json”、“application/xml”、“text/plain”三种，“application/json”和“application/xml”两种情况下可以对响应内容的结构进行进一步定义，如：响应内容为“application/json”，规定json内容里的参数类型等。
- 响应示例：响应内容的示例。
- 响应头：返回响应的Header。

返回响应也可以直接引用公共模型中已定义好的“公共响应”，并且支持自动生成响应示例。

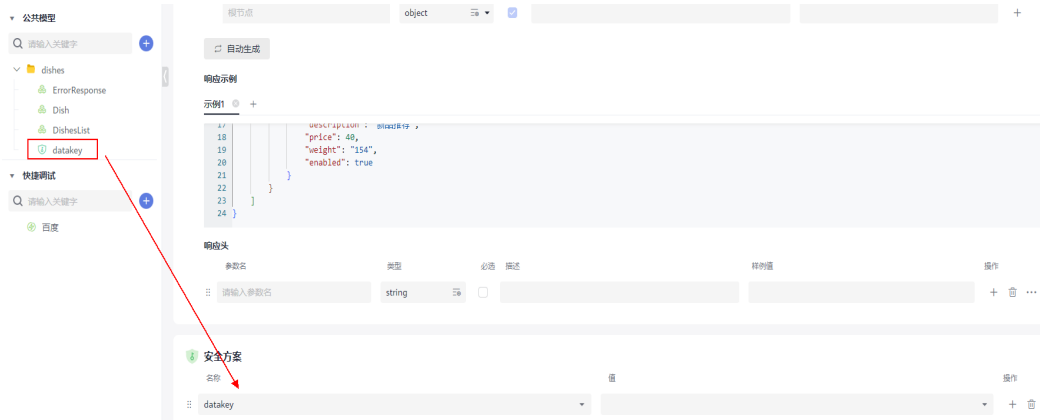


安全方案

OpenAPI规范中，安全模型对应OpenAPI3.0的components.securitySchemes。大部分的Web服务都需要经过身份认证才能访问，security就是用于描述API的安全信息和访问授权协议等信息的对象，Open API支持的最常见授权方案如下：

- API key
- HTTP
- Oauth2.0
- OpenID Connect

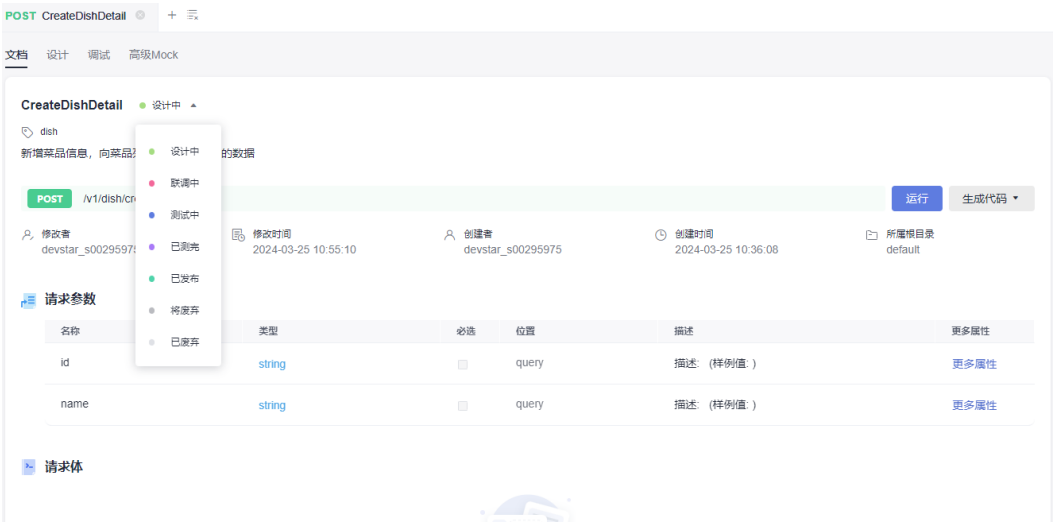
安全方案引入需要在公共模型中建立安全模型，详见**•安全模型**。



API 文档

API设计完成后，单击“保存”，会自动跳转到API文档页面。如果单击左侧导航栏中的API，默认进入API文档页面。

API文档页面展示了定义好的API信息，包括API路径、请求参数、请求体、返回响应等信息，可通过右侧文档目录切换至对应模块进行查看。在文档展示页面，可以修改API状态。单击“运行”，可切换到调试页面进行API调试。



2.2 API 审核

API设计审核为用户提供了一套成熟的审核流程，帮助检查项目中设计的API接口是否符合需求。

2.2.1 审核设置

角色权限

项目经理和项目创建者有审核设置权限。

审核功能开启

1. 单击“项目设置 > 功能设置”，进入“功能设置”页面。
2. 单击“审核设置”页签，审核设置按钮默认为关闭状态，单击，开启审核功能。

说明

开启审核功能后，项目下在API设计中创建、删除、修改API的操作均会被记录为草稿内容，需要通过审核后生效。

审核功能关闭

1. 在“审核设置”页签，单击  ，弹出“审核设置”对话框。

2. 根据提示输入“DELETE”后，单击“确定”，关闭审核功能。

说明

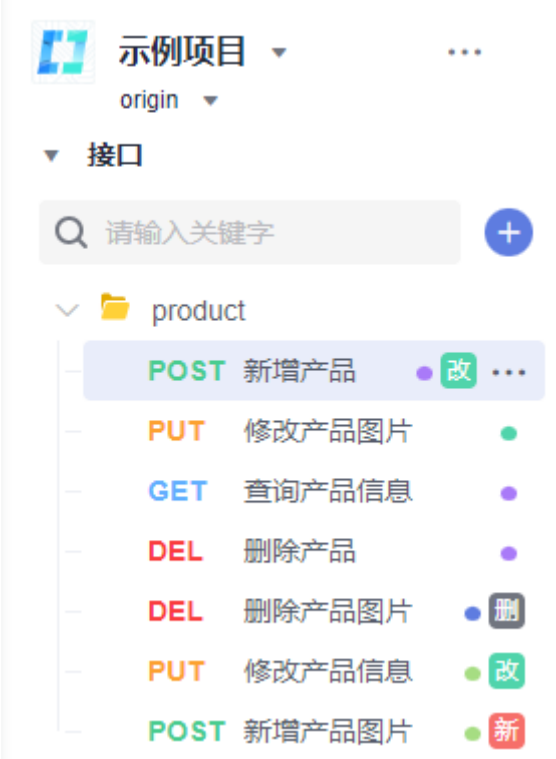
关闭审核功能后，API设计中对API的操作将立即生效，不再需要审核，同时清空未生效的API修改记录和未审批通过的提交审核记录。

2.2.2 提交审核

审核功能开启后，对于API接口的操作都需要提交审核并通过后才可生效。

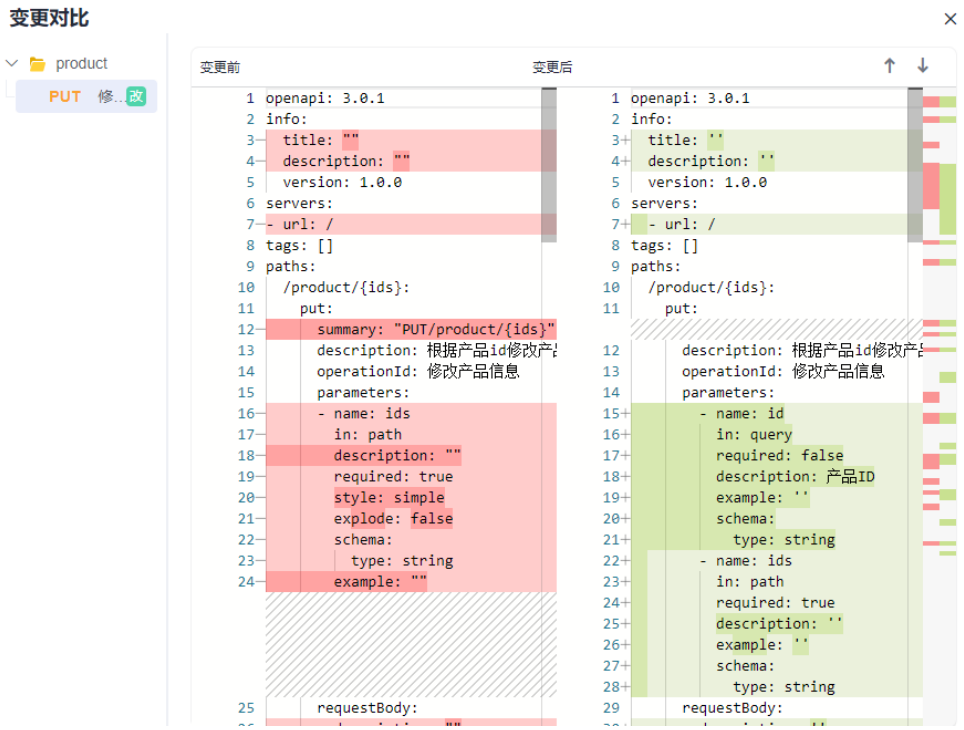
查看接口变更对比

1. 当进行新增、修改、删除API接口的操作后，接口列表会出现相应提示图标。



2. 单击变更的API接口，在API“文档”页面单击“变更对比”。

3. 弹出“变更对比”页面，可查看API接口变更前后的对比详情。



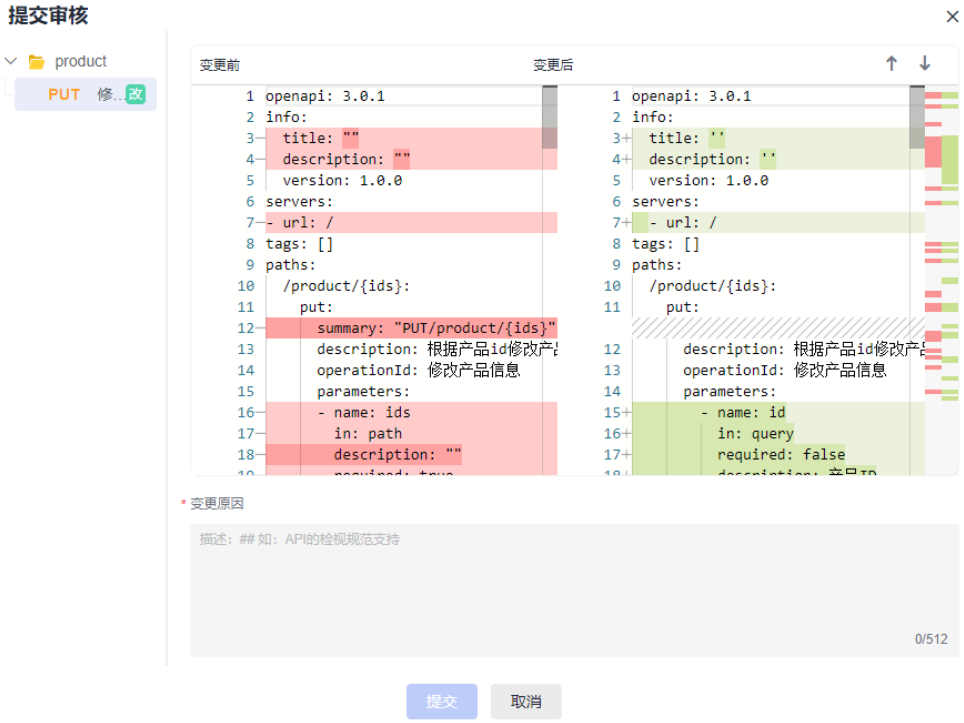
说明

其他项目参与者有权限同步查看项目下任意API接口的变更详情。

提交审核

1. 在API接口的“文档”页面单击“提交审核”，弹出“提交审核”页面，可查看接口变更前后的对比详情。

2. 填写“变更原因”，单击“提交”，提交API接口进入审核流程。

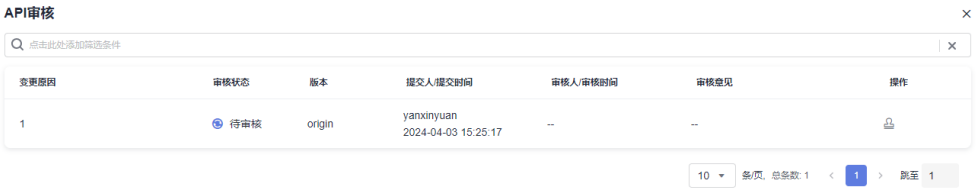


说明

审核流程结束前，该提交审核的API接口将处于锁定状态，不可编辑或删除。

撤回审核

1. 在API接口的“文档”页面单击 API审核，弹出“API审核”页面，可以查看项目中已提交审核的API接口列表。



2. 选择需要撤回的API接口，单击操作栏的，进入“审核详情”页面，可查看接口变更前后的对比详情。

3. 单击“撤回”，根据需要填写审核意见，撤回已提交审核的API接口。

2.2.3 API 审核

角色权限

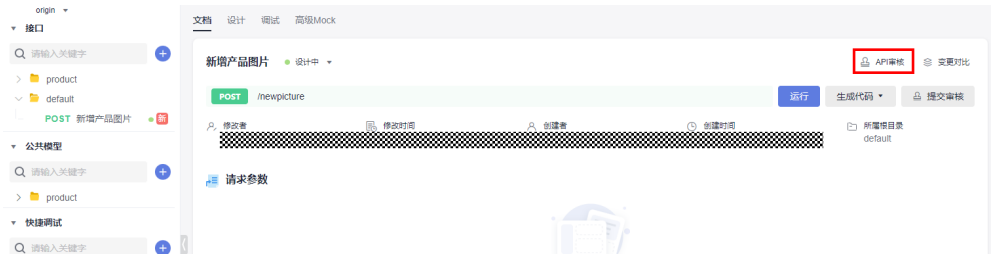
API审核操作用户角色类型及操作权限说明如下：

表 2-1

项目角色	操作	数据范围
项目创建者	查看	所有数据
	审核	所有数据
	撤回	自己提交的数据
项目经理	查看	所有数据
	审核	所有数据
	撤回	自己提交的数据
开发人员	查看	所有数据
	撤回	自己提交的数据

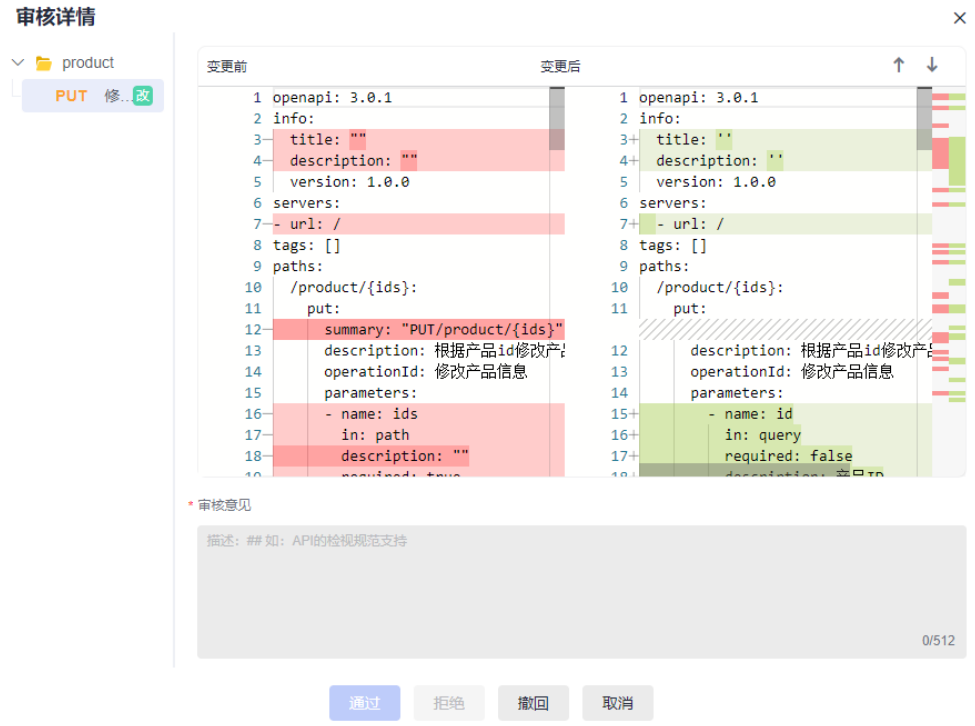
操作步骤

1. 在任意API接口的“文档”页面，单击“API审核”。



2. 弹出“API审核”页面，可查看项目中所有提交审核的API接口列表。

3. 单击操作列的  ，弹出“审核详情”页面，可查看API接口变更前后的对比详情。



4. 填写“审核意见”，单击“通过”或“拒绝”，完成API接口的审核。

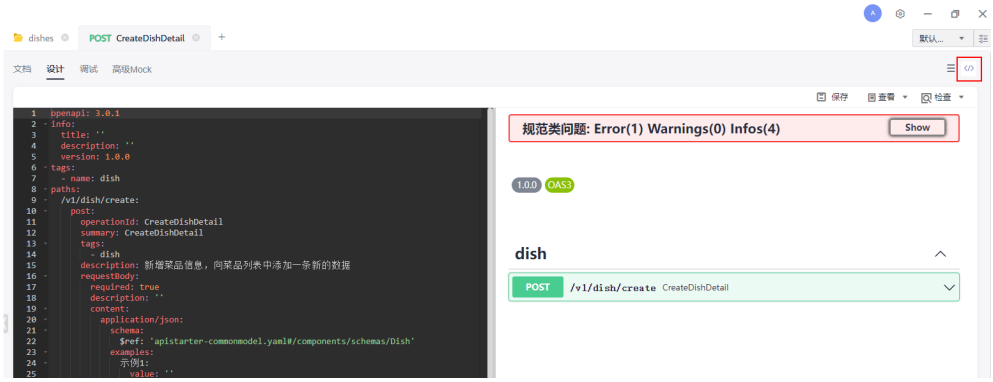
说明

对于已提交审核的API接口，提交人员可通过进入“API审核”页面进行**撤回审核**的操作，撤回时可根据需要填写审核意见。

2.3 编辑模式与 API 规范检查

CodeArts API在API设计界面提供代码编辑模式进行API设计，可使用yaml格式设计接口文档。在编辑模式中CodeArts API提供API规范性检查。

- 编辑模式



- 规范性检查

API设计规范定义，请查看**设计规范设置**。

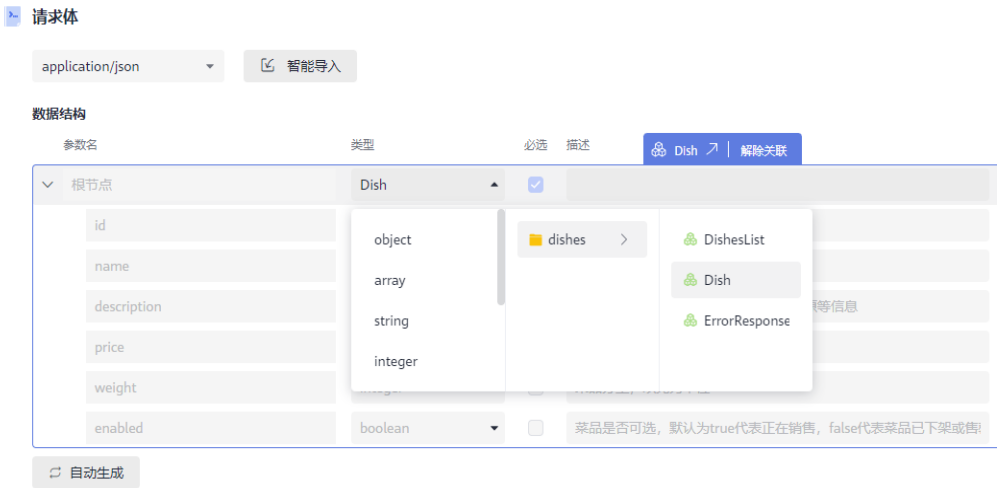
2.4 公共模型

公共模型提供API设计时所需公共数据的定义，在设计API的Body请求体或返回响应时，如果需要使用某种数据模型、公共响应等，可直接引用对应的公共模型，单击“公共模型”搜索框右侧 ，新建不同类型公共模型，CodeArts API提供以下7种公共模型定义：数据模型、公共响应、公共参数。公共请求体、公共示例、安全模型、公共响应头。

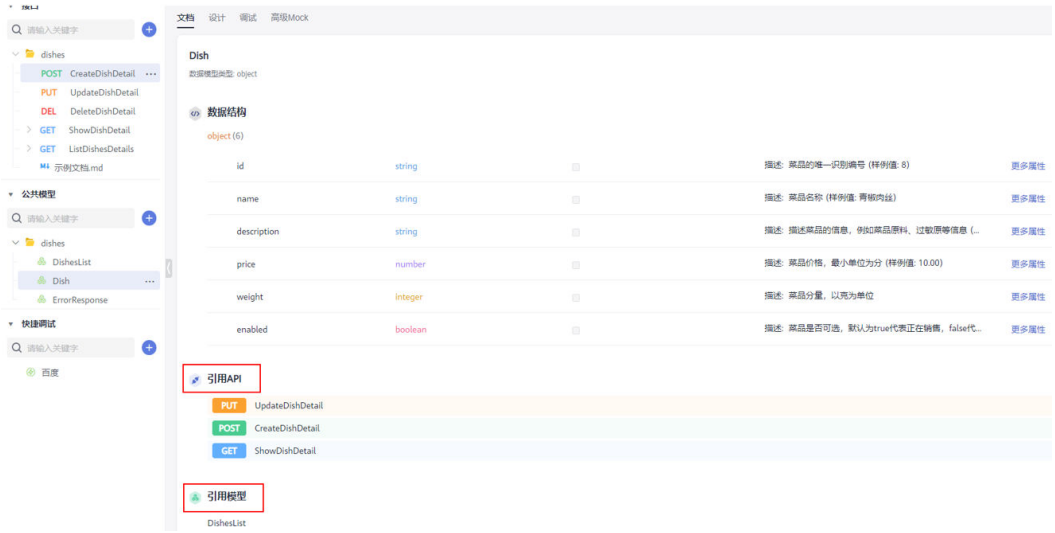
数据模型

数据模型和编程语言里的数据结构概念类似，主要用于接口设计场景中的“返回响应”和json/xml类型的“Body参数”。

在设计API的Body体或响应体时，可以直接引用公共数据结构。

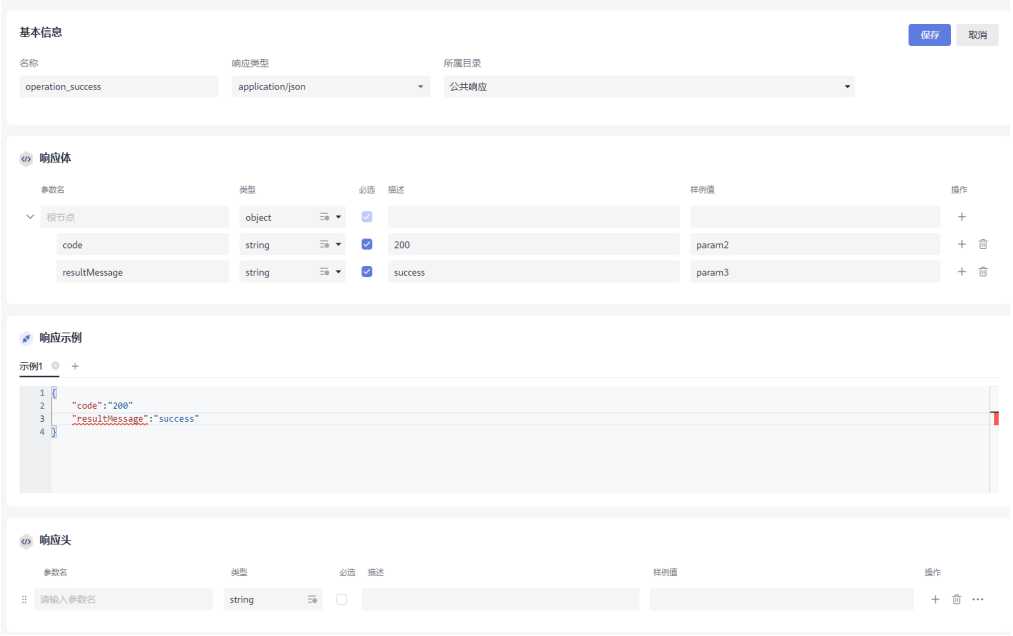


数据结构被引用后，可以在数据结构的文档页面查看引用当前数据结构的API和模型。

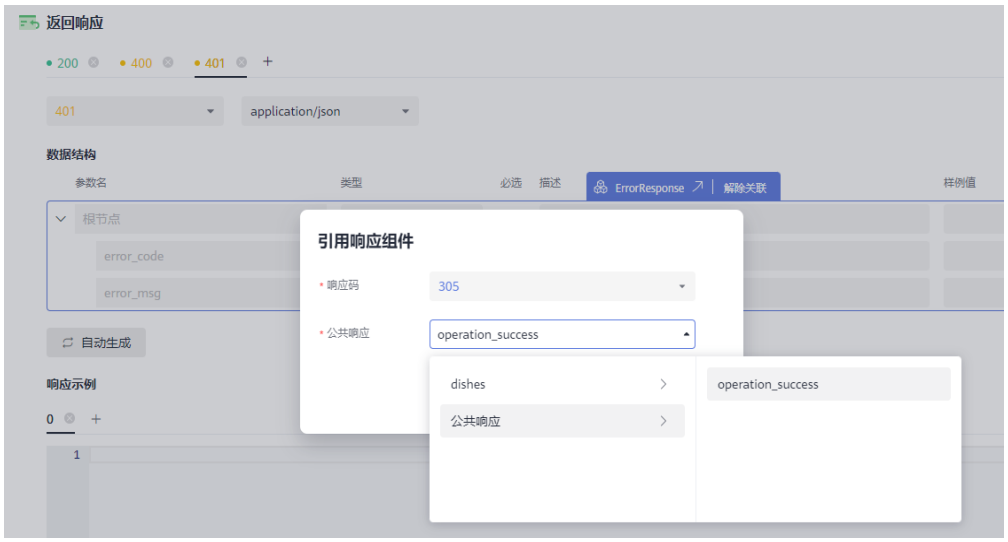


公共响应

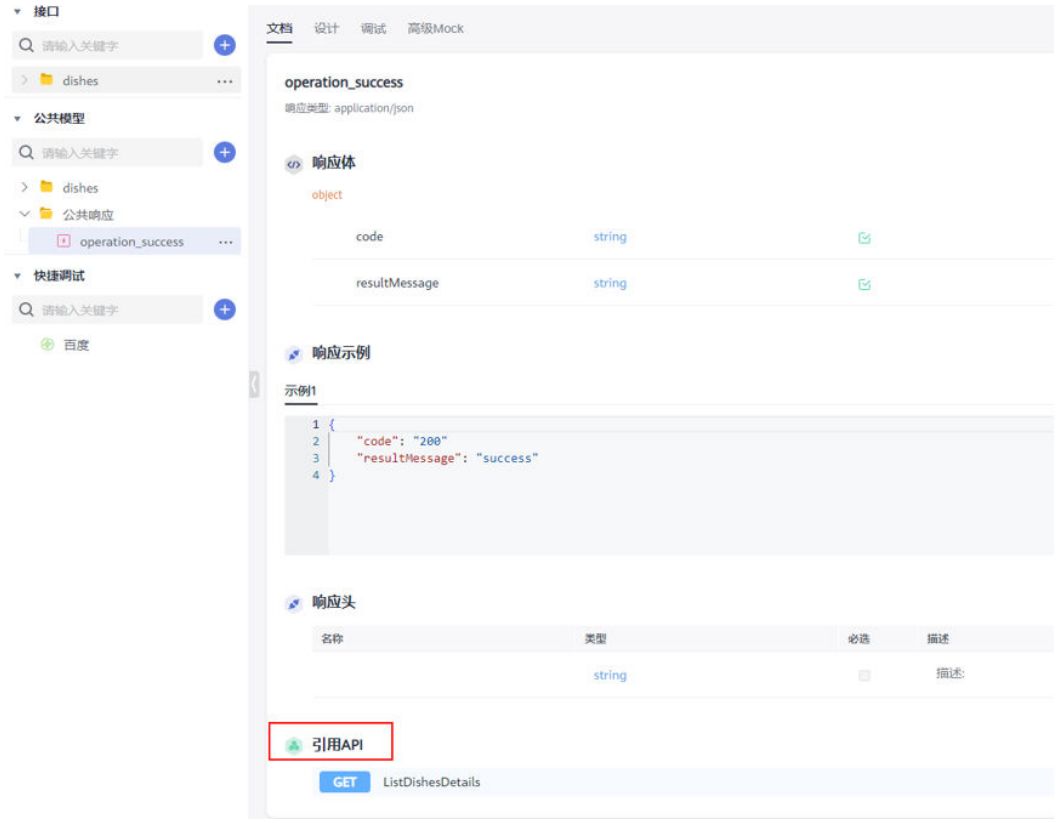
接口的公共响应返回结果，响应类型默认为“application/json”结构。



在设计API的响应体时，可以直接引用已设计好的公共响应。



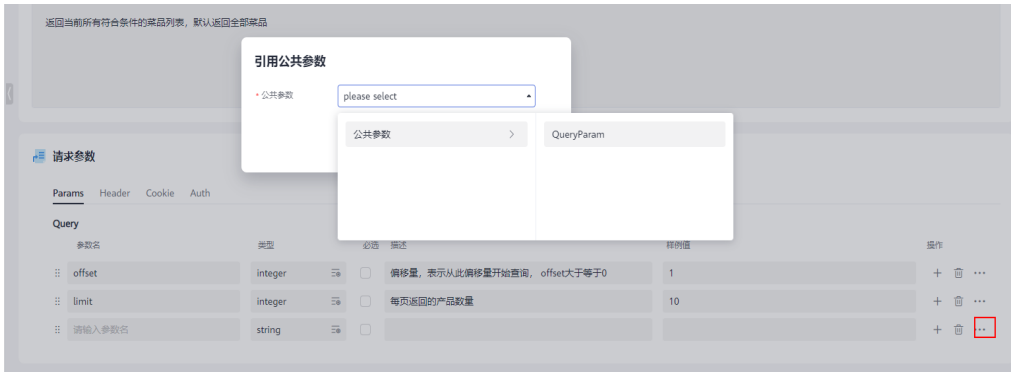
公共响应文档页面可以查看引用当前公共响应的API。



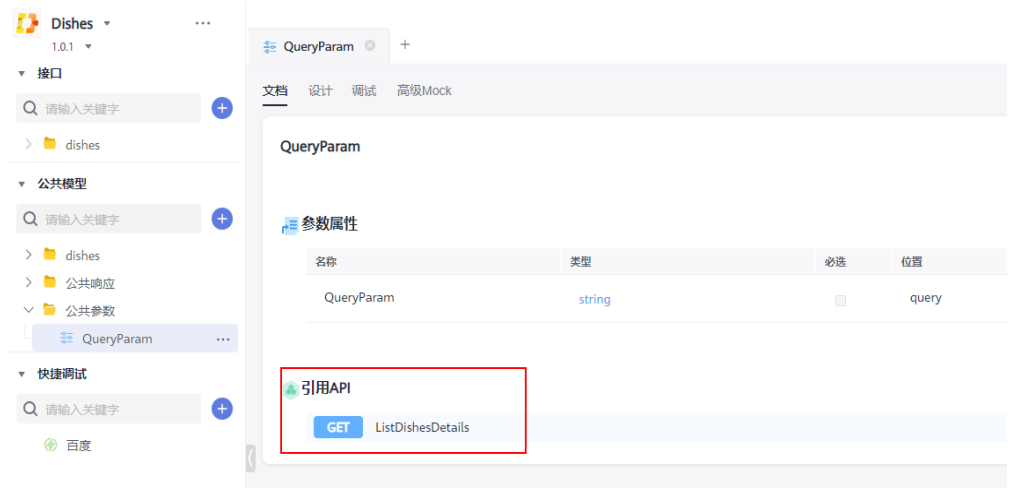
公共参数

接口的公共请求参数，参数位置可选“query”参数、“path”参数、“header”参数。

在设计API的请求参数时，可以引用已设计好的公共参数。

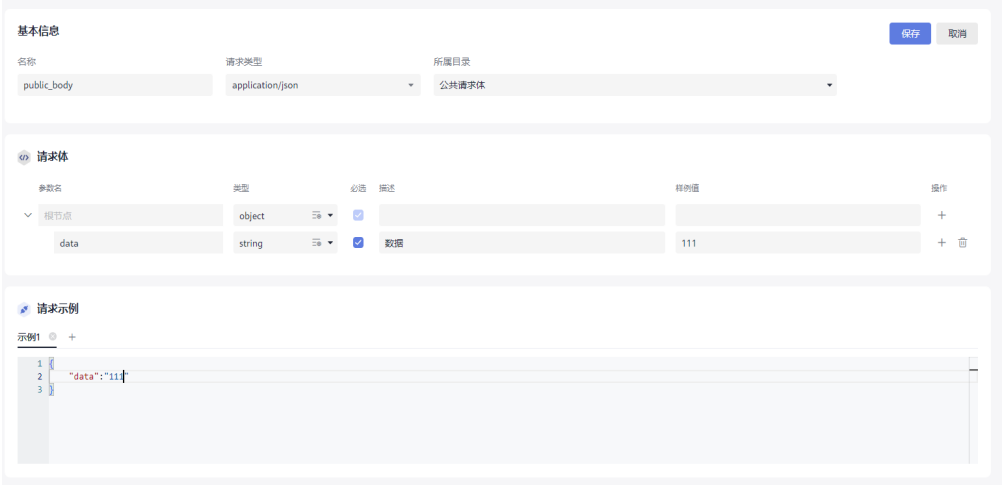


公共参数文档页面可以查看引用当前公共参数的API。

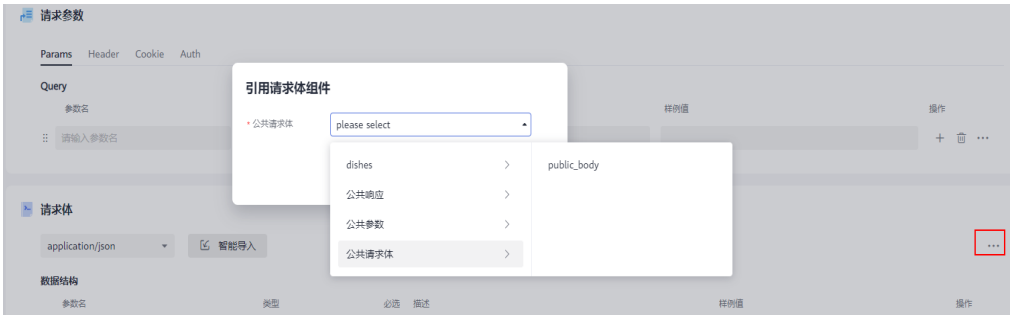


公共请求体

接口的公共请求体，默认请求类型为“application/json”。



支持设计Body体引用，并且在公共请求体的文档页面支持查看引用当前公共请求体的API。



公共示例

接口的公共示例，可在API设计请求体示例与返回响应的响应示例中被引用，在公共示例的文档页面可查看引用当前公共示例的API。

安全模型

安全模型类别提供“http”、“apikey”、“oauth2”、“openIdConnect”四种类型选择，选择不同类型的安全模型后，需要在方案内容中完善相关必填信息，用于API设计中“安全方案”的引用。在安全模型的文档页可以查看引用当前安全模型的API。

公共响应头

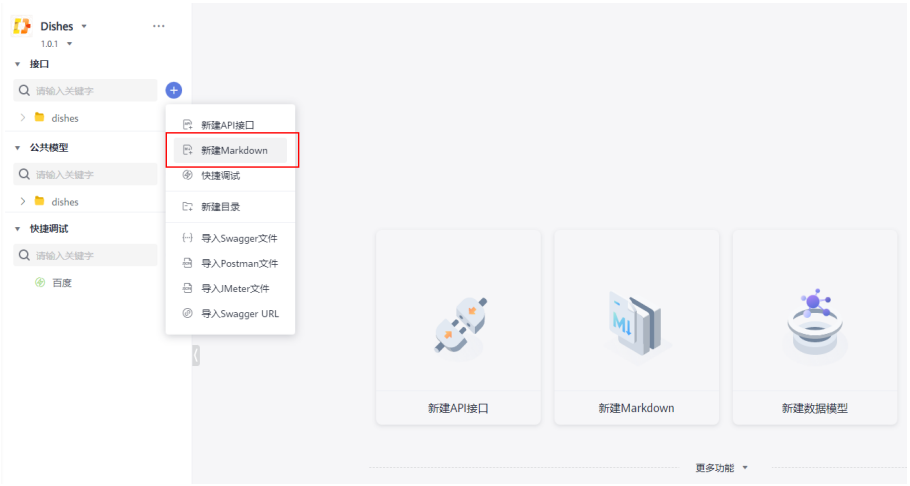
接口的公共响应头，在API设计中“返回响应”中响应头可进行公共响应头的引用。在公共响应头的文档页可以查看引用当前公共响应头的API。

2.5 Markdown 设计

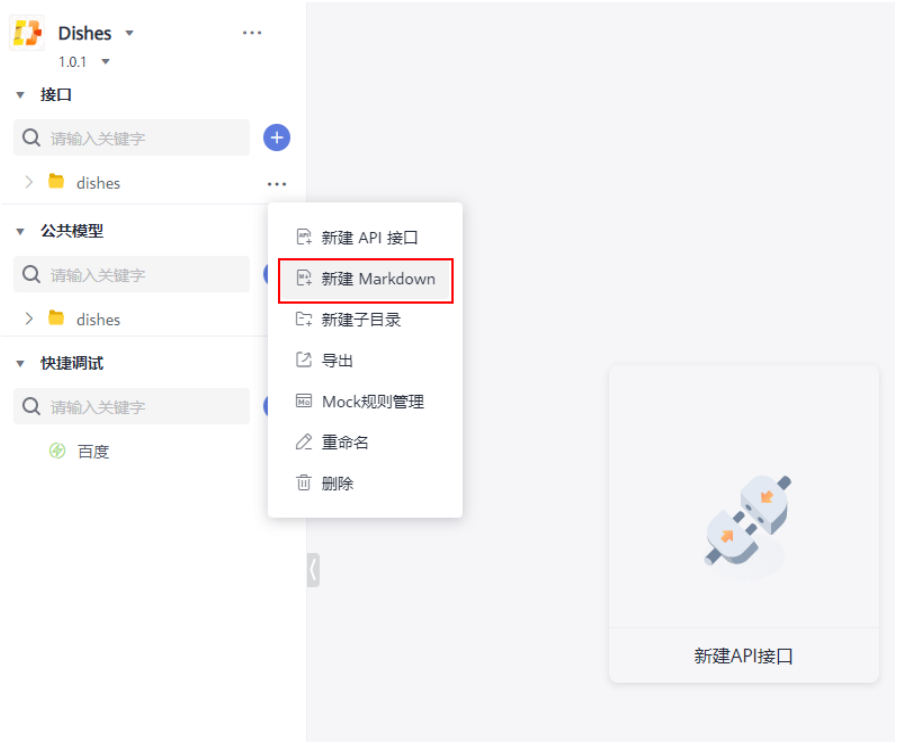
步骤1 新建Markdown，可通过以下三种方式完成。

- 方法一：单击“新建 Markdown”图标。

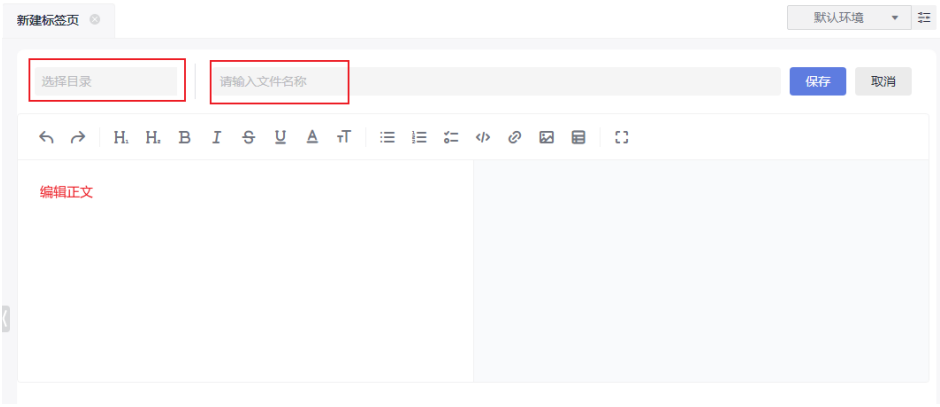
- 方法二：单击目录树搜索框右侧的 ，在弹出的下拉框中，选择“新建Markdown”。



- 方法三：在目录树中选择一个目录，单击右侧 ··· 图标，在弹出的下拉框中，选择“新建Markdown”。



步骤2 进入“新建标签页”，在该标签页可以编辑Markdown文件，主要包括：目录、名称、正文。



- 目录：在上一步中，如果通过方法一和方法二新建Markdown，目录初始内容为空，可以手动选择；如果通过方法三新建Markdown，目录会自动生成为选中的目录，也可以手动修改。
- 名称：可根据业务需要自定义命名，不能为空。命名规则：只支持中文、英文、下划线、数字和“.”，且以中英文开头，长度为3-64个字符。
- 正文：可手动输入文字，也可使用组件插入特定的内容，如：表格、超链接、图片；正文样式也可以使用组件进行调整；正文编辑支持全屏模式；在左侧编辑正文时，右侧会同步展示编辑内容，便于查看、修改样式等。

步骤3 编辑完成后，单击右上角“保存”，左侧目录树同步刷新，界面会自动跳转到文档页面，展示名称、正文，上方标签页的名称也会更新为Markdown文件名称。

步骤4 移动Markdown（可选）。

单击并拖拽Markdown到目标目录后释放即可，移动成功后，目录树会自动刷新，Markdown文件的目录也会同步改变。

说明

- Markdown文件只能在同一根目录下进行移动。
- 处于编辑页面的Markdown文件不能移动。

----结束

2.6 API 调试

完成API设计之后，在API文档页面单击“运行”会自动切换到API调试页面，也可以单击“调试”页签切换到API调试页面，本节介绍调试页面相关信息配置。

说明

由于浏览器限制，使用CodeArts API的Web版本进行接口调试，需要先安装浏览器扩展，浏览器扩展请查看**10.2 浏览器扩展**。

接口路径

API调试页面的接口路径，可以选择http或https，并在定义的路径前自动添加了域名。添加的域名是在右上角选择的环境参数中定义的，单击左侧下拉箭头可在不同运行环境间切换，单击下拉箭头右上角  按钮，可进行**环境参数配置**。

针对各个环境，默认提供变量“hostURL”参数作为添加到接口前的域名，从而拼接成完整地址。

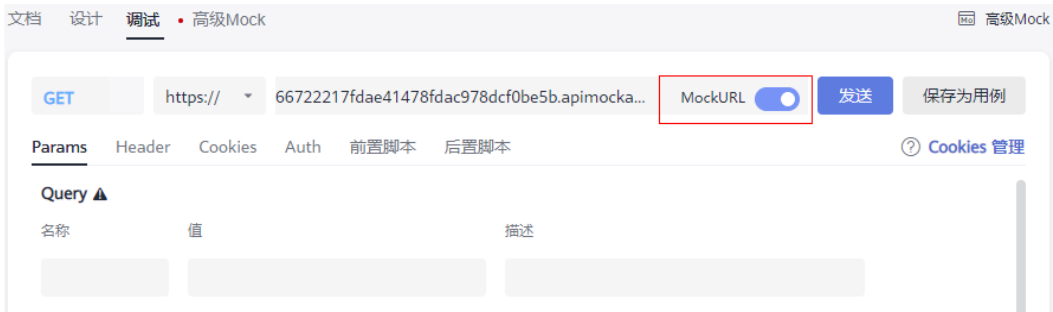
可以看到对应环境“默认环境”中的“hostURL”参数的值“127.0.0.1:8080”被添加到接口URL前，作为整个路径的域名。鼠标悬浮在路径前，提示次前置URL来自于哪一环境，单击环境名称，可直接跳转至相应环境下进行编辑。

说明

- 路径中大括号“{}”中的字符串表示Path参数。
- 路径中“?”后的字符串表示Query参数。
- 如果在调试页面路径栏下方的Query属性框中，对Query参数进行修改，路径中的Query参数也会同步变化。

Mock 选项

打开URL输入框右侧的MockURL按钮，开启Mock服务，接口路径的域名会变成云端Mock地址。Mock服务具体功能请请参考**API Mock**。



请求参数

在Params页签，可以查看以下请求参数：

- Query参数：默认填充了接口定义好了的Query参数，名称可修改。
- Path参数：默认填充了接口定义好了的Path参数，名称不可修改。
- Header参数：默认填充了接口定义好了的Query参数，名称可修改。

请求体

在Body页签中，可以查看请求体，目前支持的请求体有如下几类：

- none：无body参数。
- form-data：Content-Type为“multipart/form-data”，可以输入请求体中每个参数的类型、名称和值。
- application/x-www-form-urlencoded：Content-Type为“application/x-www-form-urlencoded”，可以输入请求体中每个参数的名称和值。
- raw：在文本框中输入文本内容，在“raw”选项旁边可以下拉选择输入文本内容格式，如Text和Json。

说明

- 请求参数的默认值均来自API设计的样例值。
- 在API调试页面修改请求参数，不能保存至API设计与API文档中。

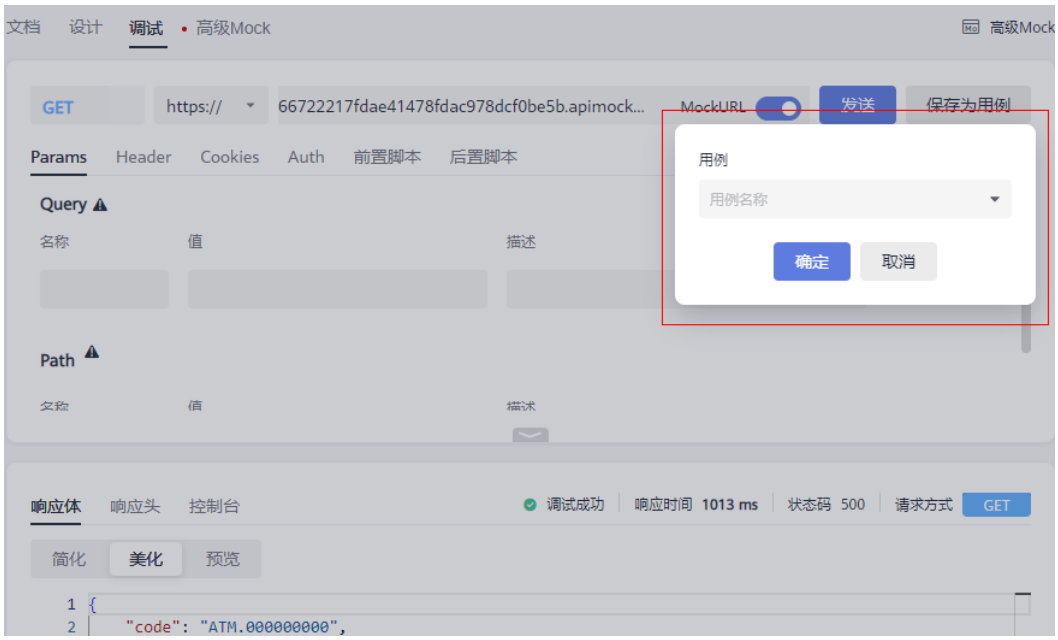
返回响应

显示响应是否成功、响应时间、状态码、请求方式以及响应体信息和响应头信息。

保存为用例

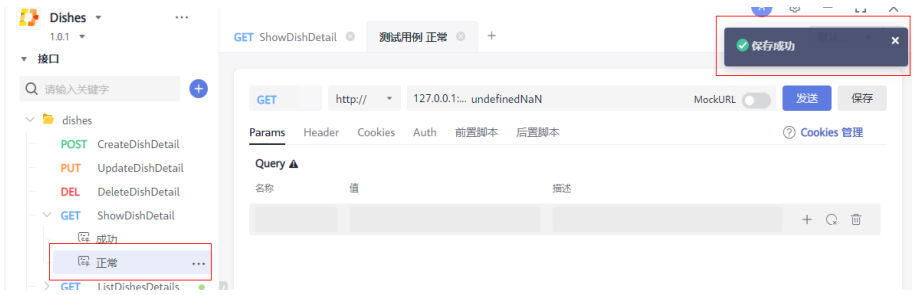
CodeArts API支持将调试结果保存为接口用例，接口用例可以记录请求参数，方便后续接口调试。

步骤1 单机发送后，在返回响应中调试成功后，单击“保存为用例”，弹出“用例”弹窗，自定义用例名称或选择系统预设名称。



步骤2 单击“确定”。

步骤3 右上角提示“保存成功”后，在左侧目标API下可查看到已保存的用例。

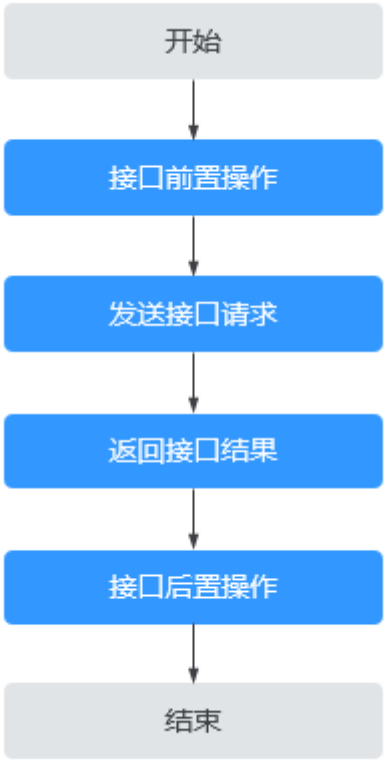


----结束

2.7 脚本能力

2.7.1 执行流程

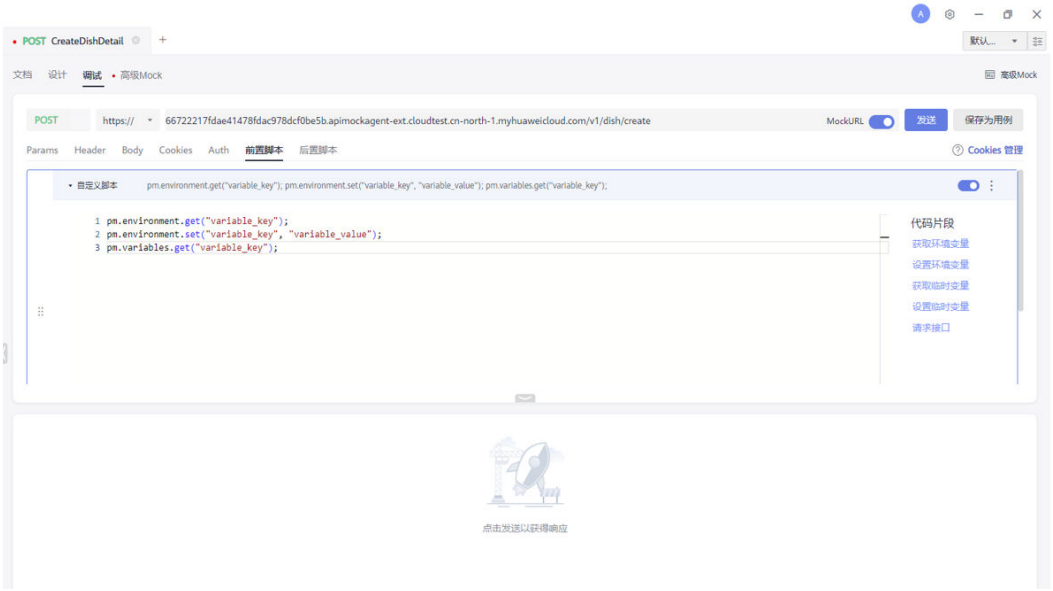
CodeArts API提供基于JavaScript的脚本引入，通过脚本可以实现在快捷调试或接口请求时添加动态行为。脚本执行流程如图：



- CodeArts API兼容Postman脚本语法，可以将Postman脚本迁移到CodeArts API中使用。
- CodeArts API脚本能力仅限于快捷调试、接口调试、接口调试用例。
- 接口的前置脚本与后置脚本支持配置多个，并按照配置顺序执行。
- CodeArts API支持脚本输入提示。

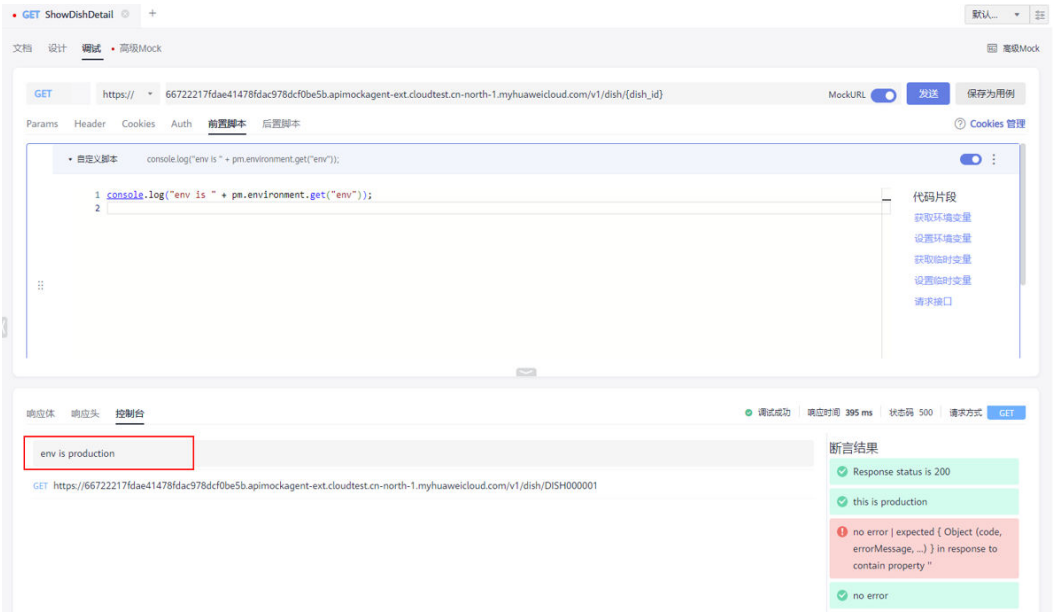
2.7.2 前置脚本

前置脚本是在请求发送前执行的代码片段。例如需要在请求头中生成时间戳、打印日志、设置随机参数值等。可以在接口的“前置脚本”页面中单击“添加前置操作 > 自定义脚本”，发送接口请求前将自动运行前置脚本。



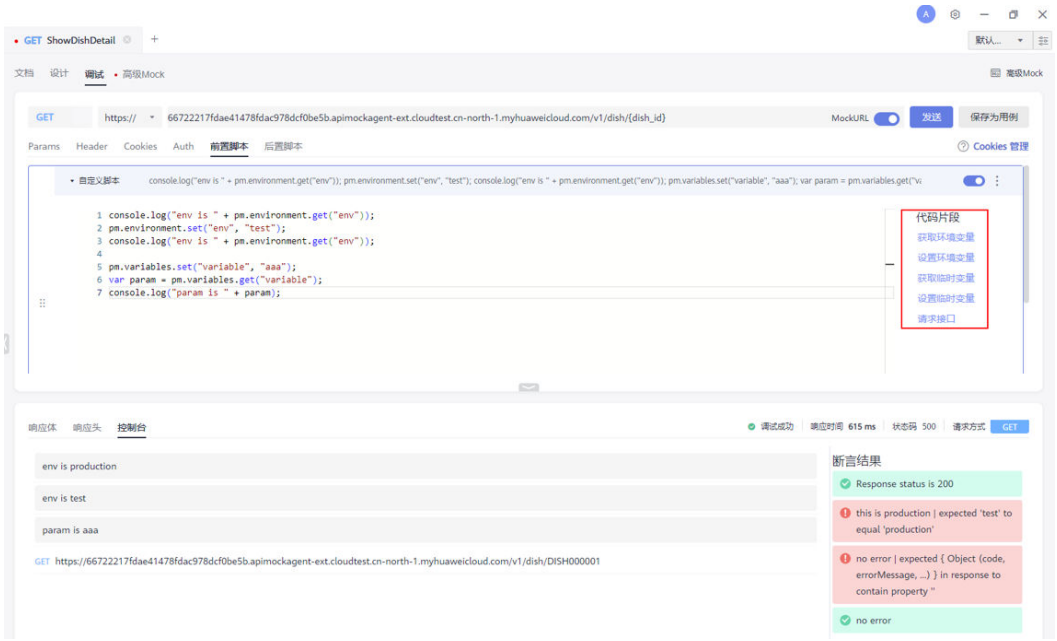
打印日志

可以通过“console.log()”将需要的变量打印在控制台，用以查看某个变量的当前值。



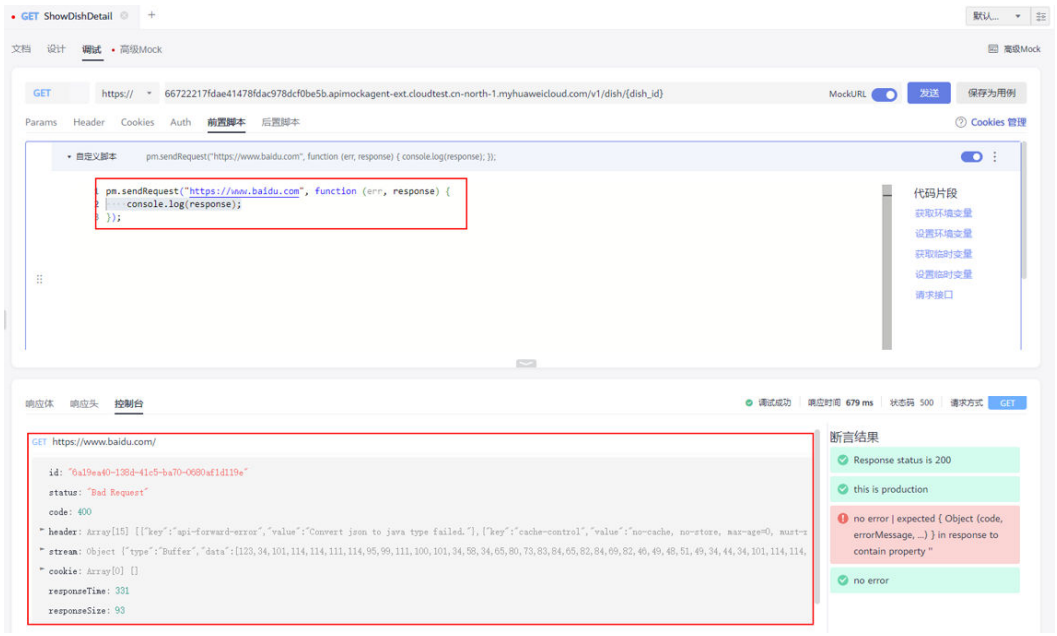
操作变量

可以通过脚本对环境变量和临时变量进行特定操作，此类操作可以通过代码片段直接引入。



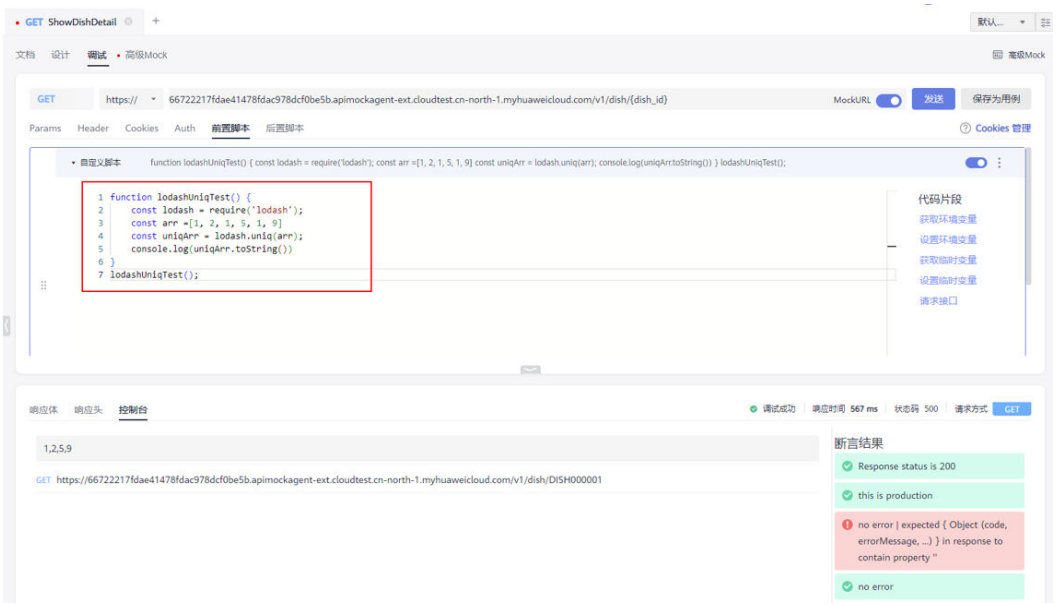
发送请求

可以通过脚本发送https请求，此操作可以通过代码片段直接引入。



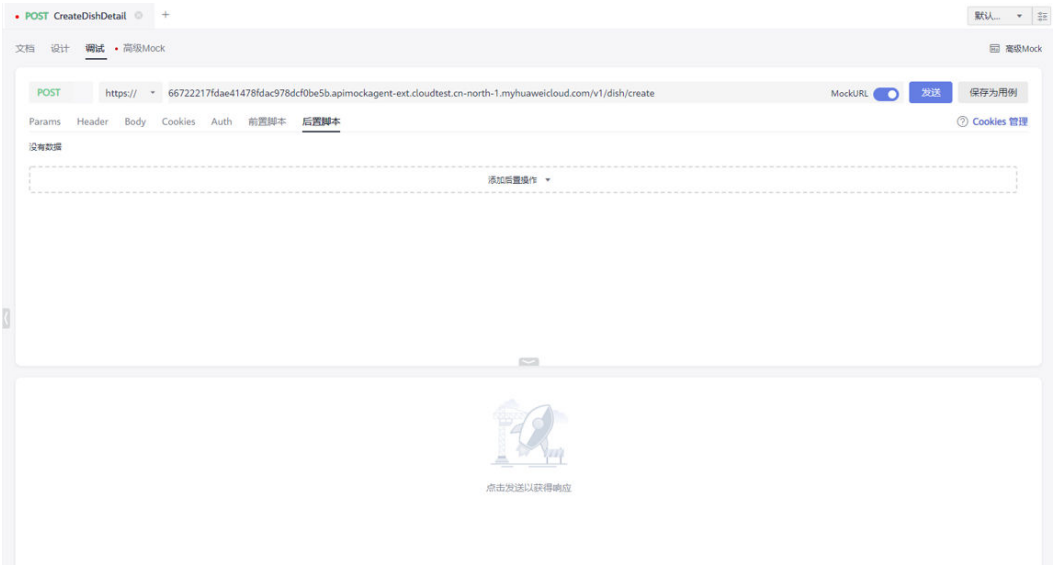
编写并执行 JS 函数

可以在脚本中定义函数，并且可调用此函数。



2.7.3 后置脚本

后置脚本是在请求发送后执行的代码片段，主要用于验证请求返回的结果（断言）、将请求返回的结果数据写入环境变量等场景。可以在接口的“后置脚本”页面中单击“添加后置操作”，发送接口请求后将自动运行后置脚本。



断言

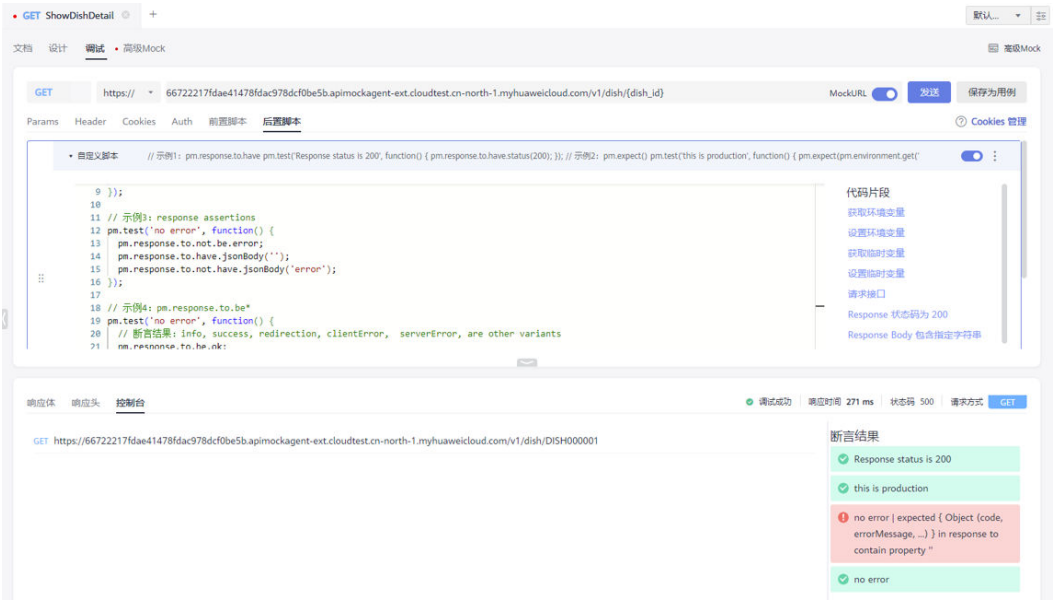
可以利用后置脚本验证API响应返回结果的正确性。

```
// 示例1：pm.response.to.have
pm.test('Response status is 200', function() {
  pm.response.to.have.status(200);
});

// 示例2：pm.expect()
pm.test('this is production', function() {
  pm.expect(pm.environment.get('env')).to.equal('production');
});
```

```
// 示例3：response assertions
pm.test('no error', function() {
  pm.response.to.not.be.error;
  pm.response.to.have.jsonBody('');
  pm.response.to.not.have.jsonBody('error');
});

// 示例4：pm.response.to.be*
pm.test('no error', function() {
  // 断言结果：info, success, redirection, clientError,  serverError, are other variants
  pm.response.to.be.ok;
  // 断言有body，并且是json格式
  pm.response.to.be.withBody;
  pm.response.to.be.json;
});
```



说明

后置脚本与前置脚本具备相同能力，包括：打印日志、操作变量、发送请求、编写并执行复杂的JS函数等。

2.7.4 pm 对象 API 参考

pm

pm:Object

pm对象，含有接口运行关联信息。可利用它获取或设置环境变量和全局变量，且访问请求信息和返回结果信息。

pm:info:Object

pm.info对象，含有接口运行关联信息。

- pm.info.eventName:String：获取当前执行的脚本类型：前置脚本（prerequest），后置脚本（test）。
- pm.info.iterationCount:Number：返回执行循环的总次数，仅集合测试有效。
- pm.info.iteration:Number：获取当前执行循环（iteration）次序，仅集合测试有效。

- pm.info.requestId:String：获取运行中接口用例的唯一ID。
- pm.info.requestName:String：获取运行中接口用例名称。

对于变量的处理

pm.variables

临时变量，**Variable SDK 参考**。

- pm.variables.toObject():function：获取所有临时变量，结果以对象形式返回。
- pm.variables.has(variableName:String):function：检查某个临时变量是否存在。
- pm.variables.get(variableName:String):function：获取某个临时变量。
- pm.variables.set(variableName:String, variableValue:String):function：设置某个临时变量的名称和值。
- pm.variables.replaceIn(variableName:String):function：用字符串替换指定变量的值，如{{variable_name}}。

pm.environment

环境变量。

- pm.environment.name:String：获取环境名称。
- pm.environment.toObject():function：获取当前环境的所有的变量，结果以对象形式返回。
- pm.environment.clear():function：清除当前环境中所有变量。
- pm.environment.has(variableName:String):function：检查某个环境变量是否存在。
- pm.environment.get(variableName:String):function：获取某个环境变量。
- pm.environment.set(variableName:String, variableValue:String):function：设置某个环境变量的名称和值。
- pm.environment.replaceIn(variableName:String):function：用字符串替换指定变量的值，如{{variable_name}}。
- pm.environment.unset(variableName:String):function：删除某个环境变量。

对于不同类型变量的优先级顺序为： 临时变量 > 环境变量。

说明

以上所有操作都是对本地值进行读写，而不会访问远程值。

对于请求的处理

pm.request

pm.request对象可以在脚本中对请求数据进行访问，**Request SDK 参考**。在前置脚本中表示即将运行的请求，在后置脚本中，表示已经运行的请求。

pm.request包含了以下结构：

- pm.request.url:**Url**：获取当前请求的URL。
- pm.request.getBaseUrl()：获取运行中环境选择的前置URL。

- pm.request.method:String：获取当前请求的方法，如GET、POST等。
- pm.request.headers:**HeaderList**：获取当前请求的headers列表。
- pm.request.body:**RequestBody**：获取当前请求的body体。
- pm.request.headers.add({ key: headerName:String, value: headerValue:String}):function：为当前请求添加指定键和值的header。
- pm.request.headers.remove(headerName:String):function：删除当前请求里指定键的header。
- pm.request.headers.get(headerName:String):function：获取请求里指定名称的header。
- pm.request.headers.upsert({ key: headerName:String, value: headerValue:String}):function：插入指定键和值的header（如果header不存在则新增，如果header已存在则修改）。
- pm.request.auth：获取请求的身份验证信息。

对于响应的处理

pm.response

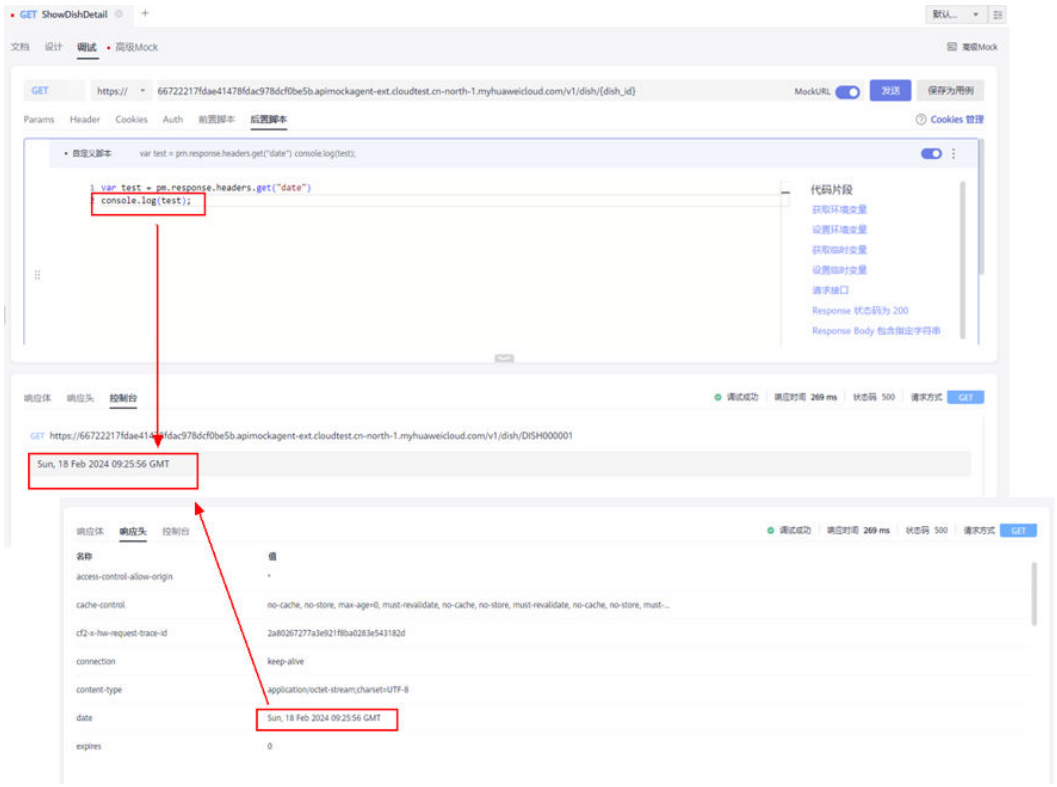
在后置脚本中pm.response接口请求完成后返回响应信息，**Response SDK 参考**。

response包含了以下结构：

- pm.response.code:Number：获取响应码。
- pm.response.status:String：获取响应状态。
- pm.response.headers:**HeaderList**：获取响应头。
- pm.response.responseSize:Number：获取响应大小。
- pm.response.text():Function：以文本形式输出响应体。
- pm.response.json():Function：以json形式输出响应体。
- pm.response.setBody('')：设置响应体。
- pm.response.headers.get：从响应头中获取指定参数的值，**Response SDK 参考**。

后置脚本中使用“pm.response.headers.get”命令可以获取响应头中指定参数的值。例如想要获取响应Header中date参数的值，那么可以在后置操作中输入如下自定义脚本：

```
var test = pm.response.headers.get("date");
console.log(test);
```

对于断言的处理

pm.test

- pm.test(testName:String, specFunction:Function):Function：断言某项结果是否符合预期。

以下示例检查返回响应是否正确：

```
pm.test("response should be OK to process", function() {
  pm.response.to.not.be.error;
  pm.response.to.have.jsonBody("");
  pm.response.to.not.have.jsonBody("error");
});
```

通过回调可选参数done，可用来测试异步方法：

```
pm.test("async test", function(done) {
  setTimeout(() => {
    pm.expect(pm.response.code).to.equal(200);
    done();
  }, 1500);
});
```

- pm.test.index():Function：获取特定位置的测试总数。

pm.expect

- pm.expect(assertion:*):Function：一个普通的断言方法，详细的说明请参照**ChaiJS expect BDD library**。该方法对于处理来自响应response或变量variables的数据断言非常有用，更多关于pm.expect断言的示例，请参见**Assertion library examples**。

Response 对象的断言API列表

- pm.response.to.have.status(code:Number)：判断响应状态码是否为设置的(code:Number)。
- pm.response.to.have.status(reason:String)：判断响应状态码是否符合响应码描述。
- pm.response.to.have.header(key:String)：判断是否包含响应头中对应的键。
- pm.response.to.have.header(key:String, optionalValue:String)：判断是否包含响应头中对应的键、值。
- pm.response.to.have.body()：判断是否有响应体。
- pm.response.to.have.body(optionalValue:RegExp)：判断响应体中是否等于设置的正则表达式。
- pm.response.to.have.body(optionalValue:String)：判断响应体中是否等于设置的字符串。
- pm.response.to.have.jsonBody()：判断响应体是否是有效的json结构。
- pm.response.to.have.jsonBody(optionalExpectEqual:Object)：判断响应体是否等于json结构里设置的对象。
- pm.response.to.have.jsonBody(optionalExpectPath:String)：判断响应体是否包含json结构里设置的路径。
- pm.response.to.have.jsonBody(optionalExpectPath:String, optionalValue:*)：判断响应体是否包含json结构里设置的路径，且路径是否等于设置的值。
- pm.response.to.have.jsonSchema(schema:Object)：判断响应体是否满足定义的json模型。
- pm.response.to.have.jsonSchema(schema:Object, ajvOptions:Object):判断响应体是否满足定义的json模型、校验配置。

pm.response.to.be

pm.response.to.be是一系列内置规则用于实现快速断言。

- pm.response.to.be.info：检查状态码是否为1XX。
- pm.response.to.be.success：检查状态码是否为2XX。
- pm.response.to.be.redirection：检查状态码是否为3XX。
- pm.response.to.be.clientError：检查状态码是否为4XX。
- pm.response.to.be.serverError：检查状态码是否为5XX。
- pm.response.to.be.error：检查状态码是否为4XX或5XX。
- pm.response.to.be.ok：检查状态码是否为200。
- pm.response.to.be.accepted：检查状态码是否为202。
- pm.response.to.be.badRequest：检查状态码是否为400。
- pm.response.to.be.unauthorized：检查状态码是否为401。
- pm.response.to.be.forbidden：检查状态码是否为403。
- pm.response.to.be.notFound：检查状态码是否为404。
- pm.response.to.be.rateLimited：检查状态码是否为429。

对于发送请求的处理

pm.sendRequest

pm.sendRequest:Function用于脚本异步发送HTTP/HTTPS的请求。

- 更多参数信息请查阅**Collection SDK 文档** 。
- 在前置脚本和后置脚本都可以使用。
- 更多参考:
  - **Request JSON 结构**
  - **Response 结构**

```
// 使用字符串URL示例
pm.sendRequest("https://postman-echo.com/get", function(err, resq) {
  if (err) {
    console.log(err);
  } else {
    pm.variables.set("variable_name", "new_value");
  }
});

// 完整请求示例
const echoPostRequest = {
  url: "https://postman-echo.com/post",
  method: "POST",
  header: {
    header_name1: "value1",
    header_name2: "value2",
  },
  // body 为 x-www-form-urlencoded 格式
  body: {
    mode: "urlencoded", // 此处为 urlencoded
    // 此处为 urlencoded
    urlencoded: [
      { key: "account", value: "codeartsapi" },
      { key: "password", value: "123456" },
    ],
  },
  /*
  // body 为 form-data 格式
  body: {
    mode: 'formdata', // 此处为 formdata
    // 此处为 formdata
    formdata: [
      { key: 'account', value: 'codeartsapi' },
      { key: 'password', value: '123456' }
    ]
  }
  // body 为 json 格式
  header: {
    "Content-Type": "application/json", // 提醒：header中需要添加Content-Type
  },
  body: {
    mode: 'raw',
    raw: JSON.stringify({ account: 'codeartsapi', password:'123456' })
  }
  // body 为 raw 或 json 格式
  body: {
    mode: 'raw',
    raw: 'body内容'
  }
  */
};
pm.sendRequest(echoPostRequest, function(err, resq) {
  console.log(err ? err : resq.json());
});

// 对返回结果进行断言
pm.sendRequest("https://postman-echo.com/get", function(err, resq) {
  if (err) {
    console.log(err);
```



```
  }
  pm.test("response should be OK to process", function() {
    pm.expect(err).to.equal(null);
    pm.expect(resq).to.have.property("code", 200);
    pm.expect(resq).to.have.property("status", "OK");
  });
});
```

2.7.5 使用 JS 类库

内置类库列表

断言

- chai（v4.2.0）：用于断言BDD/TDD断言。

加密解密库

- jsrsasign（10.3.0）：用于RSA加密/解密。

Encode、Decode 库

- crypto-js（v3.1.9-1）：编码/解码库，基本包含常用的编码、解码方式，如Base64、MD5、SHA、HMAC、AES等等。

  require方法只允许使用整个模块，不能单独使用类库里的某个子模块，具体看本文档末尾说明。
- atob（v2.1.2）：用于Base64解码。
- btoa（v1.2.1）：用于Base64编码。
- tv4（v1.3.0）：用于校验JSONSchema。
- xml2js（v0.4.19）：用于XML转JSON。

JSONSchema校验库

- ajv（v6.6.2）：校验JSONSchema。

实用工具

- postman-collection（ v3.4.0）：Postman Collection库。
- cheerio（v0.22.0）：jQuery的子集。
- lodash （v4.17.11）：JS实用工具库。
- uuid：生成UUID。
- moment（v2.22.2）：日期处理库 (不含 locales)。
- mockjs：模拟数据生成，拦截Ajax请求。
- csv-parse/lib/sync（ v1.2.4）：处理CSV格式数据。
- iconv-lite：用于字符编码转换，支持数十种字符编码格式的转换。

内置NodeJS模块

- path：处理文件路径。
- assert：提供一组断言测试。
- buffer：处理二进制数据。
- util：实用函数的集合。
- url：解析URL字符串。

- **querystring**：处理URL，查询字符串。
- **punycode**：字符编码方案。
- **string-decoder**：将Buffer对象解码为字符串。
- **stream**：处理流数据。
- **events**：处理事件。
- **timers**：在给定的毫秒数后调用函数。

通过require方法可以直接使用CodeArts API内置的JS类库。

```
var cryptoJs = require("crypto-js");
console.log(cryptoJs.SHA256("Message"));
```

使用方式

```
/**
 * 示例一：该示例引入加密算法模块 crypto-js，并使用其中 AES 对 message 进行加解密
 * @param message 需要加密的文本
 * @param key 秘钥
 */
function cryptoJSTest(message, key) {
    const CryptoJS = require('crypto-js');
    // 加密
    const ciphertext = CryptoJS.AES.encrypt(message, key).toString();
    // 解密
    const bytes  = CryptoJS.AES.decrypt(ciphertext, key);
    const originalText = bytes.toString(CryptoJS.enc.Utf8);
    console.log(originalText);
}
cryptoJSTest('test message', 'secret key');

/**
 * 示例二：该示例引入 uuid 模块生成一段 uuid，并引入 btoa 模块进行 Base64 编码
 */
function uuidBtoaTest() {
    const uuid = require('uuid');
    const btoa = require('btoa');
    const id = uuid.v4();
    const base64EncodedId = btoa(id);
    console.log(id);
    console.log(base64EncodedId);
}
uuidBtoaTest();

/**
 * 示例三：该示例引入 lodash 模块，并测试了其中的 uniq 对数组去重
 */
function lodashUniqTest() {
    const lodash = require('lodash');
    const arr =[1, 2, 1, 5, 1, 9]
    const uniqArr = lodash.uniq(arr);
    console.log(uniqArr.toString())
}
lodashUniqTest();
```


3 导入与导出

3.1 项目导入

CodeArts API提供友商项目导入功能，目前支持API Fox项目中的API设计导入CodeArts API中。

导出数据

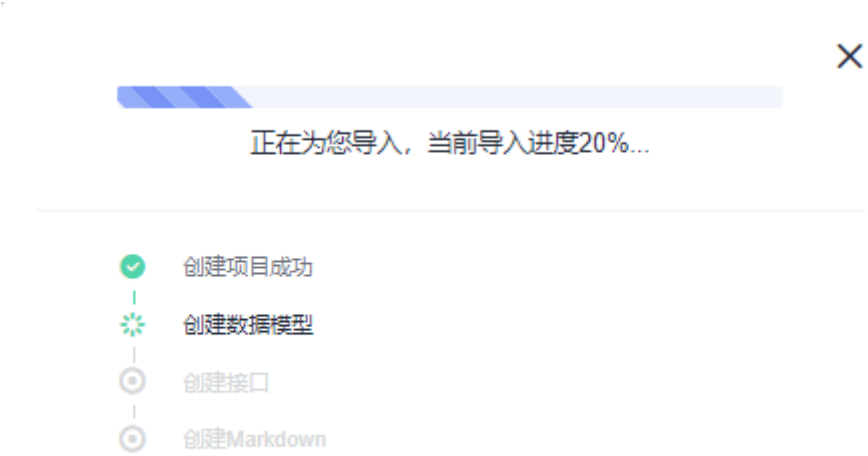
1. 打开API Fox对应项目，单击“项目设置 > 导出数据”，进入“导出数据”页面。
2. 选择“Apifox”数据格式，单击”导出”，完成导出。

导入数据

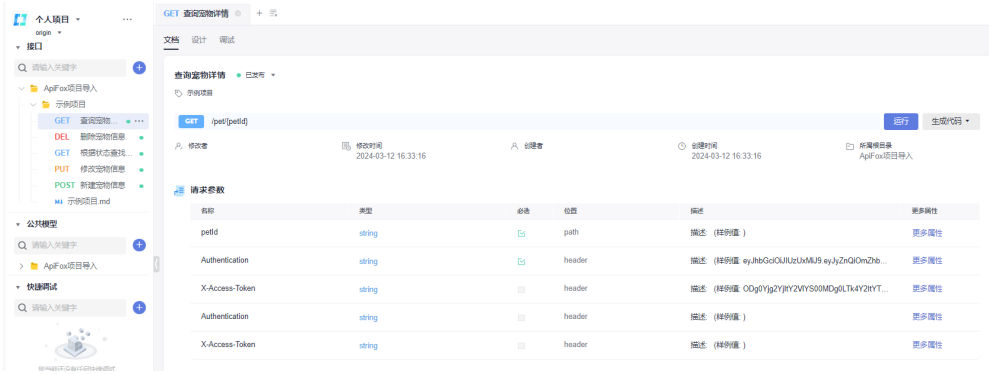
1. 打开CodeArts API首页，单击“导入项目”。

2. 弹出“导入项目”对话框，选择需要导入的Apifox文件。
3. 弹出“导入预览”对话框，选择“接口”或“数据模型”，可预览所有导入数据，然后单击“导入”。

4. 弹出导入进度对话框，可查看导入进度。



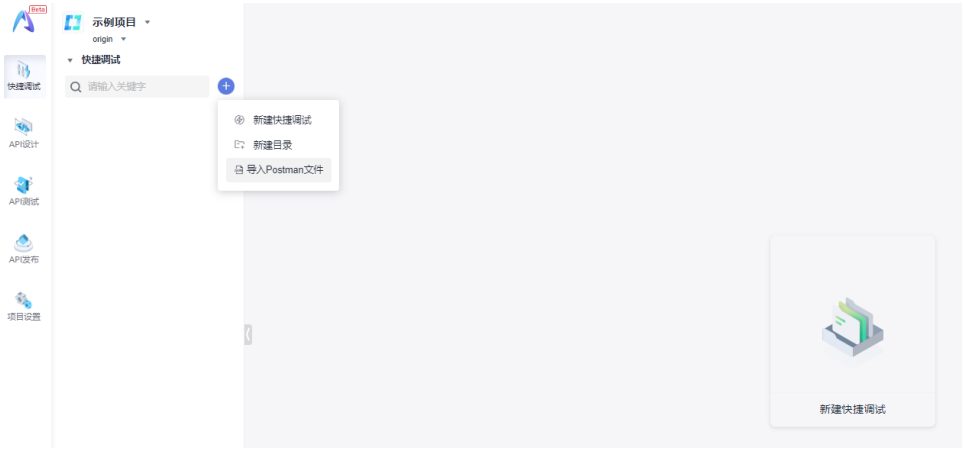
5. 导入完成后，可进入导入项目，查看导入的接口和数据模型。



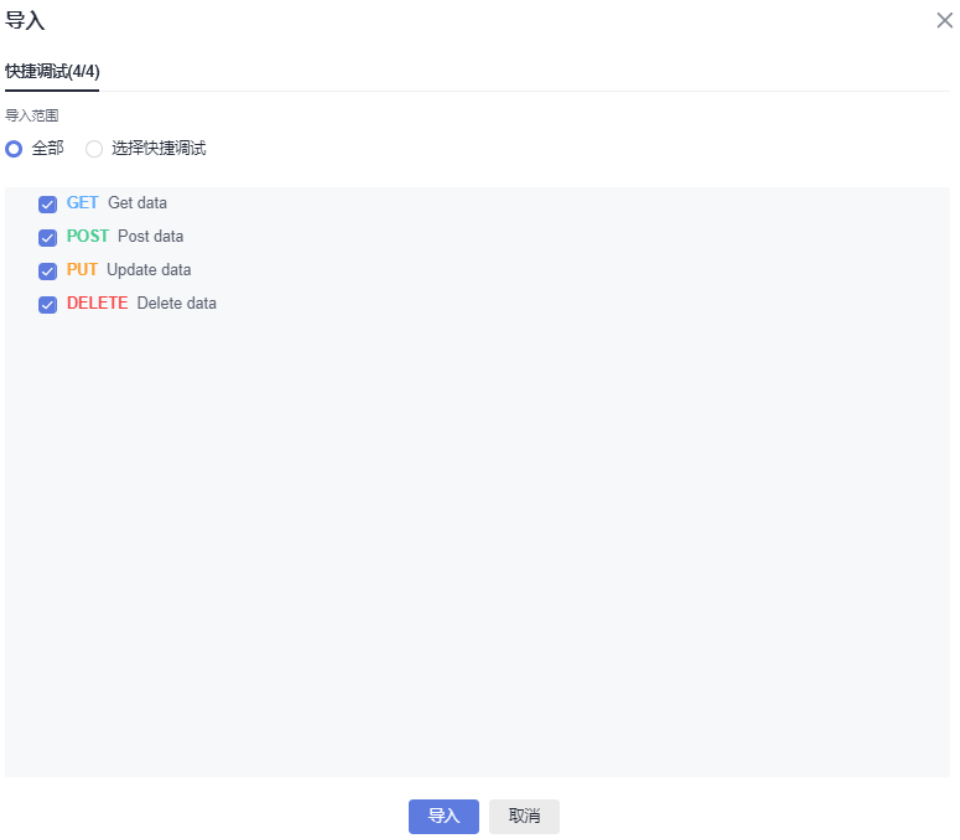
3.2 快捷调试导入

CodeArts API支持Postman的Collections数据文件的导入，方便用户迁移调试数据。

1. 进入“快捷调试”页面，单击，选择“导入Postman文件”。



2. 弹出“导入Postman文件”对话框，选择需要导入的Postman的Collection文件。
3. 弹出“导入”对话框，查看导入接口信息，单击“导入”完成数据迁移。



说明

目前仅支持单个Collection文件的导入，不支持全量Collections和Environment数据集ZIP包的全量导入。

3.3 接口导入/导出

3.3.1 接口导入

功能说明

CodeArts API支持导入以下API数据格式文件，方便数据迁移。

- Swagger
- Postman（支持Collections和Environment数据集ZIP包）
- Jmeter
- Swagger URL

手动导入

通过主页面的“更多功能”选项或单击左上角搜索框旁边的 ，可以展示当前支持的导入选项，可根据需要选择。

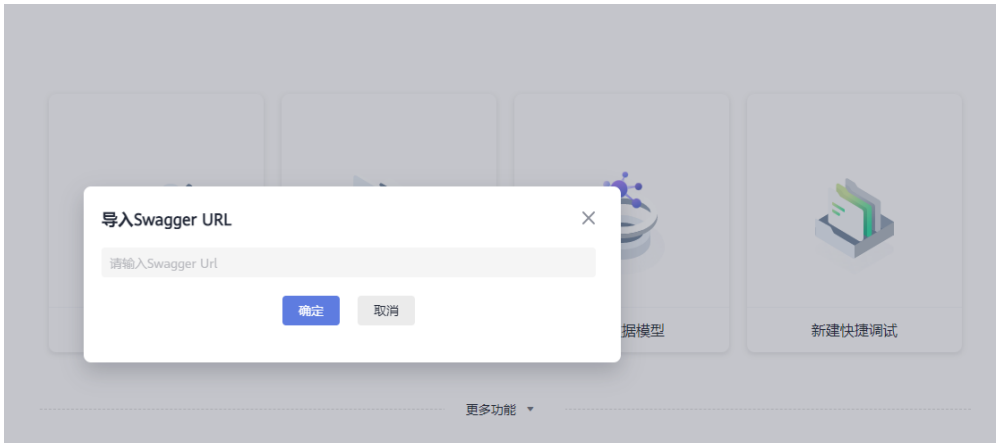
- 导入文件

  选择响应格式的文件并上传，然后选择文件中系统识别出来的接口和数据模型导入。

说明

- 一个文件导入后，将会视为一个根目录，一个根目录中不允许存在URL相同的接口。
- 文件重复导入，则新接口默认覆盖原接口。
- Postman支持Collections和Environment数据集ZIP包的全量导入，也支持单个Collection文件导入。

- 导入URL
  选择“导入Swagger URL”，输入URL，即可完成文档导入。



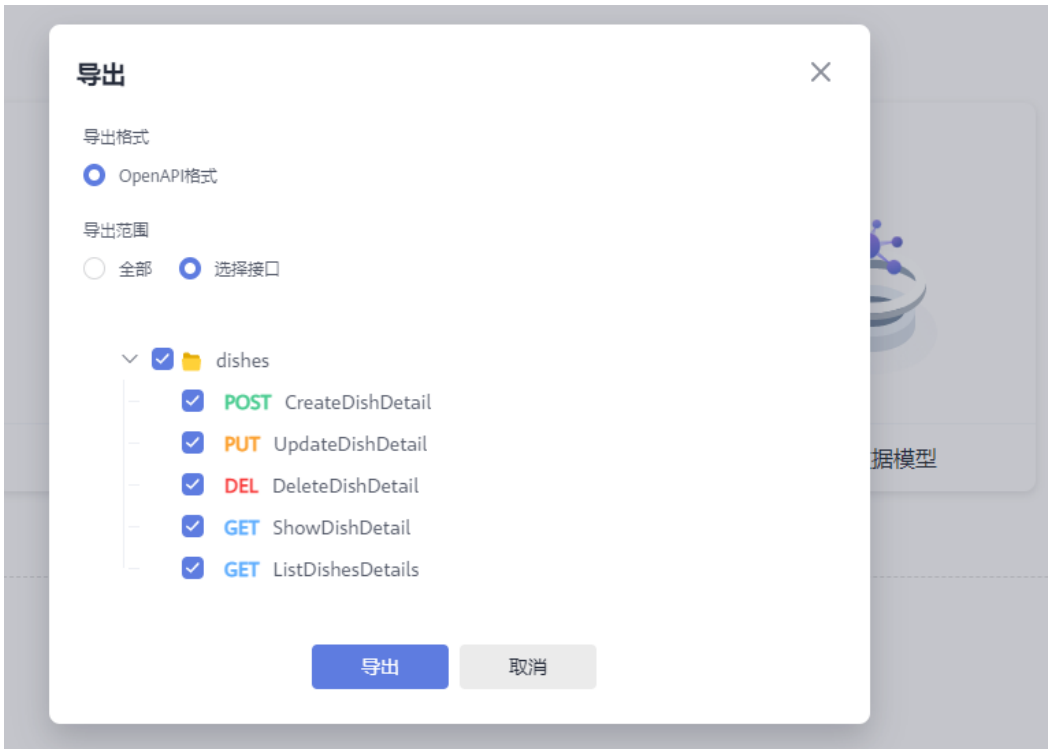
说明

通过Swagger URL导入时，需要填写json数据文件的URL，而非Swagger UI的URL。

3.3.2 接口导出

在接口目录或已设计好的接口功能目录中，可以导出OpenAPI格式的接口文件。

在弹出的窗口中，默认勾选了当前目录对应的接口，也可以重新勾选想要导出的接口。



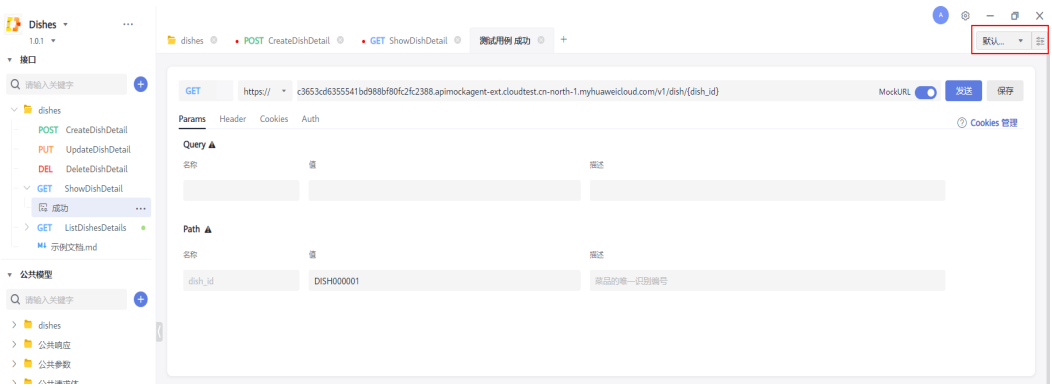
选择导出目录后，单击“导出”即可。

4 环境与环境变量

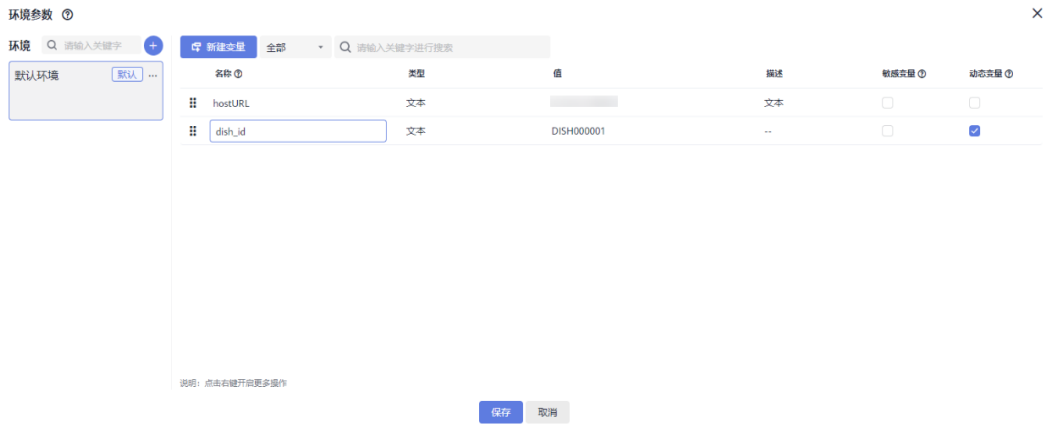
在开发项目中，不同阶段对应不同环境，例如：前端与后端的开发人员需要在研发环境做功能联调，测试人员需要在测试环境测试，上线生产环境前也需要在预发环境验证，不同环境对应不同的前置URL和接口参数值，为了方便快速切换运行环境，快速修改前置URL等参数，CodeArts API提供了环境管理功能。

环境参数入口

进入目标项目后，在右上角提供快速切换环境下拉框，提供快速搜索功能。



单击下拉框右侧  ，即可进入“环境参数”页面。



新建环境

在“环境参数”页面，单击  ，填写环境名称和描述，然后单击“确定”，即可添加一个环境。

新建变量

在“环境参数”页面，单击 

，即可添加一个变量，根据需要填写变量信息：

参数项	说明
名称	当前环境变量参数名，可使用“$${参数名}”形式进行引用。
类型	包括：文本、随机字符串、随机整数、时间戳、格式化时间戳、生成UUID、Base64编码、MD5哈希值、密码或认证信息、SHA512编码。
值	当前环境变量的值，支持加密显示。
描述	根据不同类型，自动添加默认描述，也可根据设计需要自定义描述。
敏感变量	勾选为敏感变量后，云测对参数值加密存储，在测试结果日志中使用星号覆盖处理。敏感数据类型适用并不限于个人信息、鉴权信息等，如：姓名、地址、用户名等。
动态变量	动态参数的值可以在用例执行过程中被赋值。动态参数初始值可为空，被赋值之后，此处显示的是最新值。 动态参数赋值方法：在用例测试步骤“响应提取”的“赋值给环境动态参数”中设置后，测试执行时，响应提取的内容将被赋值给动态参数。

变量搜索

CodeArts API支持在当前环境下搜索变量，可以根据如下条件搜索：

- 变量参数选择：全部、名称、值、描述、动态变量。
- 关键字：支持关键字快速搜索。

参数使用

环境参数可以用于整个项目，如：测试步骤的参数、检查点、变量、URL等都可以引用环境参数。

环境参数的引用形式为“$${参数名}”，如：参数名为“hostname”，则可以使用“$${hostname}”来引用该参数。可以按环境上下文配置环境参数，如：类生产环境中hostname=stage.example.com，生产环境中hostname=prod.example.com。测试执行时可选择环境执行。

5 API 开发

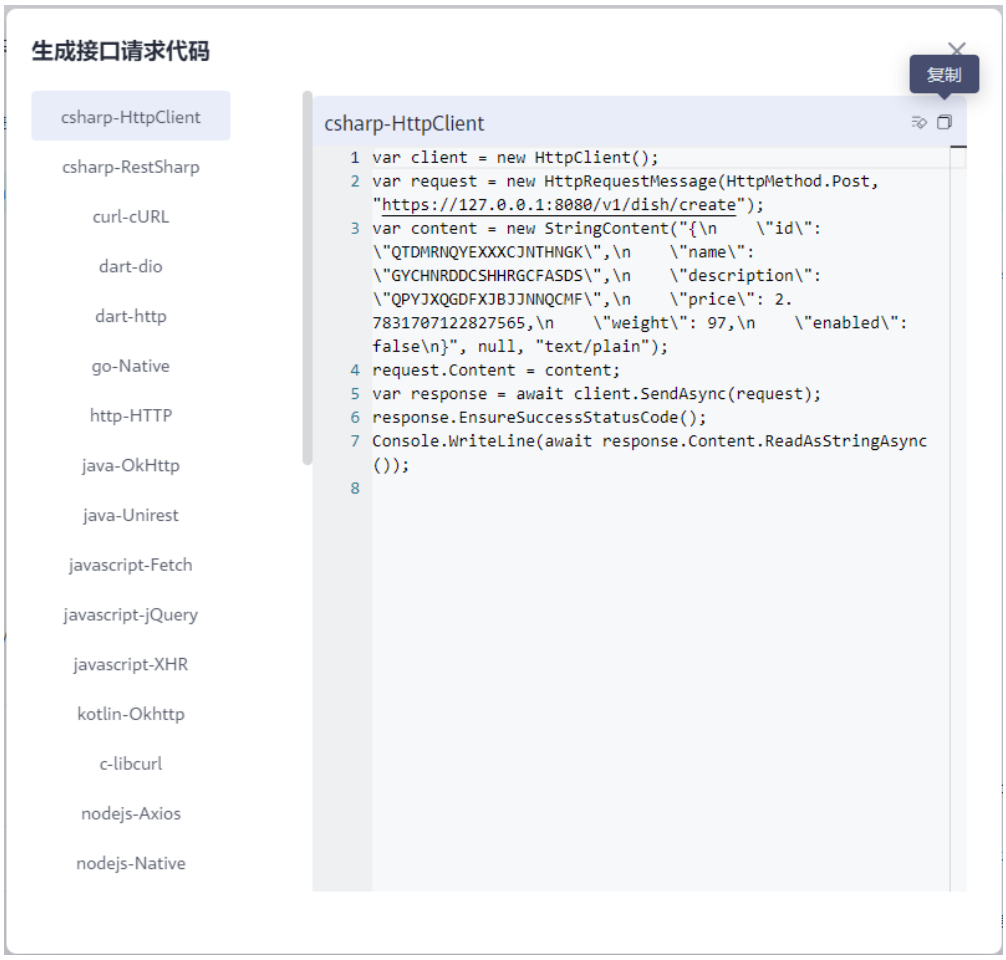
CodeArts API支持生成接口请求代码和业务代码，缩短前后端开发时长，规范代码结构。

基于接口定义，CodeArts API支持自动生成各种语言和框架的请求代码和业务代码。在API设计模块，接口文档界面右侧，即可看到“生成代码”的入口。

- 生成接口请求代码

  单击“生成代码”下拉框，选择“生成接口请求代码”，弹出“生成接口请求代码”对话框。

  支持生成多种语言的接口请求代码，单击右上角，可一键复制并直接粘贴，支持编辑修改和代码格式化。



- 生成业务代码

  单击“生成代码”下拉框，选择“生成业务代码”，弹出“生成业务代码”对话框。

  根据需要选择语言、生成范围（包括“仅当前接口及关联模型”和“整个项目”），单击“生成”将生成的代码存储到本地，开发者可以将代码复制到本地的研发工具（如IntelliJ IDEA）。

6 API Mock

6.1 背景介绍

随着微服务系统及分布式架构的发展，多模块或子系统的依赖为软件系统的开发与测试带来了很多挑战。比如在测试某个模块时，依赖的第三方服务不能返回想要的结果，或第三方服务不稳定时，该模块的测试进度则会受到影响。

Mock服务能很好解决这些问题，使用场景如下：

- 并行开发：在被依赖模块未开发完成时，使用Mock服务替代真实业务服务场景，可进行并行开发。
- 依赖服务不稳定：当依赖服务不稳定时，会导致自动化测试用例失败，使用Mock服务替代真实服务，可以保证自动化测试稳定执行，提升流水线的健壮性。
- 构造异常场景：测试时需要构造某些异常数据或延迟响应等异常场景，真实服务通常无法满足需求，使用Mock服务替代可以快速构造异常场景，提升测试覆盖率。
- 前后端分离开发：前端开发往往依赖于后端数据接口，在后端接口就绪之前，前端往往很难推进，前端可以通过Mock功能制造场景数据接口来进行开发与调试。

6.2 Mock 规则管理

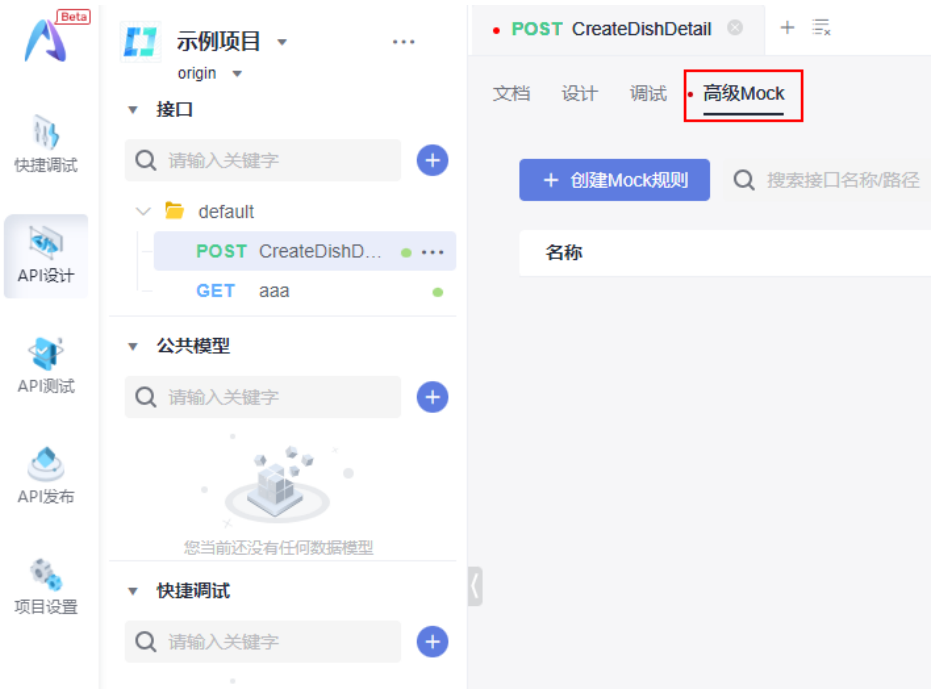
规则入口

可通过以下两种方式进入Mock规则管理页面。

- 入口一
  a. 单击“API设计”，然后单击接口目录右侧的···，选择“Mock规则管理”。

b. 进入“Mock规则管理”页面，可以查看当前目录下所有接口和接口对应的Mock规则。

- 入口二
  a. 单击“API设计”，选择对应的接口。
  b. 单击“高级Mock”页签，进入“Mock规则管理”页面，可以查看该接口对应的Mock规则。



创建规则

在“Mock规则管理”页面，单击“创建Mock规则”，可以创建接口下的Mock规则。设计并创建一个Mock规则，请参见**Mock规则设计**。

搜索规则

“Mock规则管理”页面的接口支持模糊搜索： 输入接口名称或路径名的关键字，即可搜索对应接口。

自定义匹配优先级

在匹配规则的时候，按照从上到下的顺序依次进行匹配；在匹配到符合条件的规则时，返回当前规则的响应内容。

接口下的规则默认按照创建时间匹配优先级，支持通过手动拖拽方式自定义匹配优先级。单击要调整的规则，并拖动到对应位置，松开鼠标，即可完成优先级的调整。

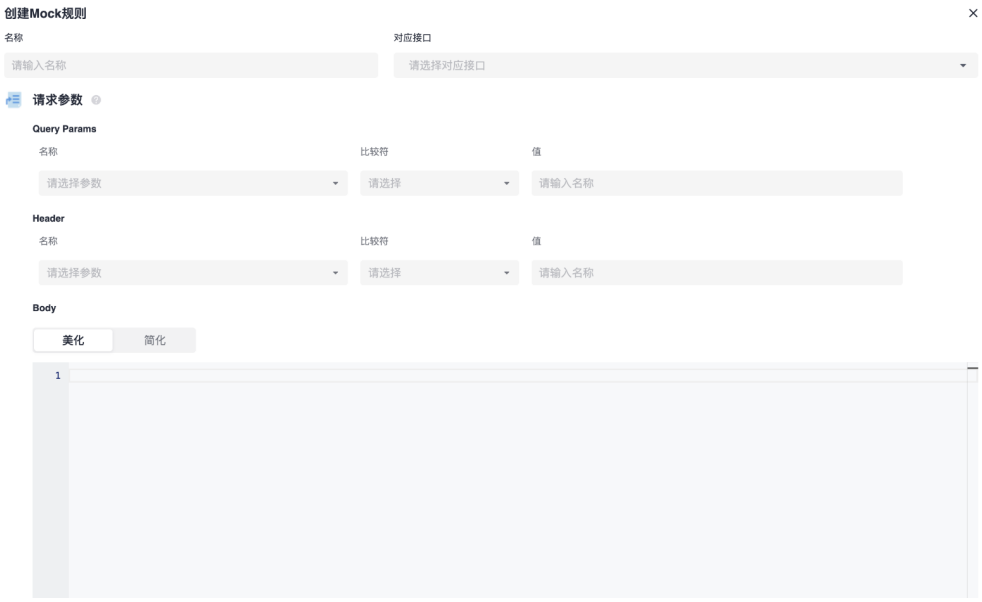
6.3 Mock 规则设计

配置请求规则

- 请求Mock服务时，系统会根据规则中配置的参数匹配，只有规则下所有参数满足匹配条件时，才可匹配到整个规则。
- 请求参数，支持Query、Path、Header、Body。

  说明

  Path参数只有在当前选择的接口的url中存在path变量时才可以显示配置。
- Body支持json和xml格式。当请求体为json格式时，匹配规则会同时匹配json中的key和value。



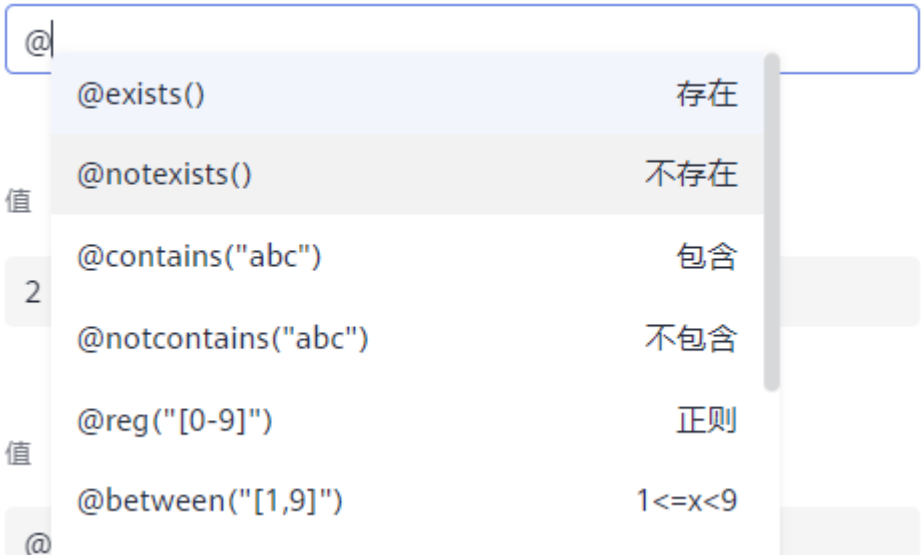
说明

无法保存重复规则。如果同一个接口下的不同规则配置内容一致，会提示无法保存规则。

复杂规则匹配

Query、Path、Header支持复杂规则匹配，包括以下类型：

- <、<=、>、>=，支持类型：数字。
- =、!=，支持类型：数字、字符。
- 表达式（选择表达式后输入“@”符开启选择）。
  - 存在、不存在：判断Query、Header名称是否存在。
  - 包含、不包含：判断Query、Header、Path中的某个字段的值是否包含某个值。
  - 正则：判断Query、Header、Path中的某个字段的值是否符合正则表达式中的匹配规则。
  - 区间范围：判断Query、Header、Path中的某个字段的值是否在该区间范围内。

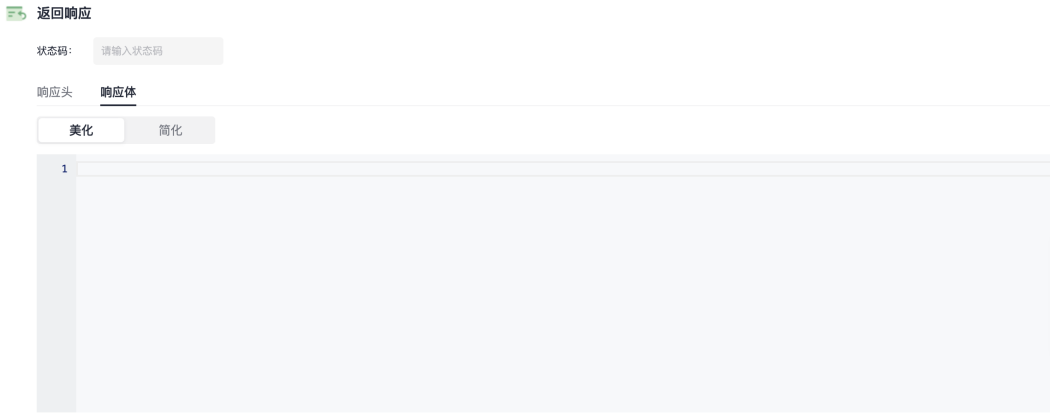


说明

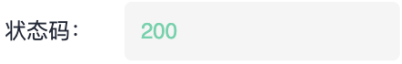
- Path不支持“存在”和“不存在”类型。
- 表达式支持Mock.js语法，具体使用方式可参考**Mock语法说明**。

返回响应

Mock服务支持从状态码、响应头、响应体三个维度配置请求的响应。



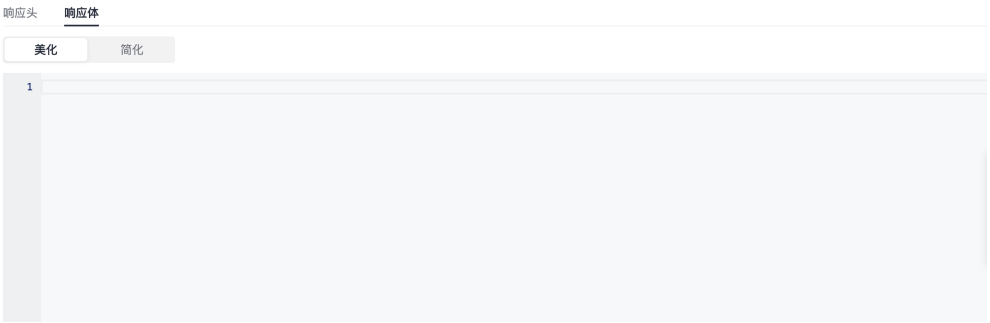
- 状态码：必填项，参考http状态码。

- 响应头：选填项，输入响应头的名称和值，配置希望返回的响应头。

- 响应体：选填项，支持json、xml及自定义文本格式。

6.4 Mock 语法说明

APIMock语法完全兼容Mock.js，并扩展了一些Mock.js没有的语法（uuid、正则表达式），Mock语法详见表6-1。

如现有Mock语法无法满足需求，建议使用正则表达式“@regexp”来实现灵活定制，正则表达式基本能满足各种特殊场景的需求。

表 6-1 Mock 语法

类型	函数
Basic	boolean、natural、integer、float、character、string、range、date、time、datetime、now、timestamp。
Name	first、last、name、cfirst、clast、cname。

类型	函数
Web	url、domain、email、ip、tld。
Address	region、province、city、county。
Text	paragraph、sentence、word、title、cparagraph、csentence、cword、ctitle。
Color	color。
Other	guid, id,uuid。
Helper	capitalize、upper、lower、pick、shuffle。
Reg	regex。

基本写法

- 支持json。函数需要写在vaule值位置，以“@”作为起始符。
- 支持xml。（本版本不支持，下个版本支持。）
- 不支持单独@函数。
- 只支持在响应体内使用mock.js。

实例如下：

输入	示例结果

{
"id":@uuid(),
"flag":@boolean(1, 9, true),
"name":@string("teststart",3,7),
"createtime":@now()
}
	{
	"id":d29925a288404207b60161fdaed05d55
	"flag":false,
	"name":"stseee",
	"createtime":2022-08-11 15:16:20
	}

基础类型

分类	规则	示例	示例结果
布尔值	@boolean	@boolean()	false，true
	@boolean(min,max,current)	@boolean(1,9,true)	false，true
自然数	@natural	@natural()	1350447356
	@natural(min)	@natural(10000)	819989031
	@natural(min,max)	@natural(60,100)	63
整数	@integer	@integer	1128800169
	@integer(min)	@integer(10000)	29427959

分类	规则	示例	示例结果
	@integer(min,max)	@integer(60,100)	94
浮点数	@float	@float()	13425418.3
	@float(min)	@float(0)	1800545158.8
	@float(min,max)	@float(60,100)	98.63203
	@float(min,max,dmin)	@float(60,100,3)	69.882191
	@float(min,max,dmin,dmax)	@float(60,100,3,5)	80.14955
单字符	@character	@character()	"7"
	@character(pool)	@character("lower")	"x"
		@character("upper")	"R"
		@character("number")	"6"
		@character("symbol")	"#"
		@character("aeiou")	"i"
字符串	@string	@string()	"%#Vwj"
	@string(length)	@string(5)	"^16)1"
	@string(pool,length)	@string("lower",5)	"zrcmq"
		@string("upper",5)	"PFTFP"
		@string("number",5)	"96480"
		@string("symbol",5)	"#&!!("
		@string("aeiou",5)	"uoauu"
	@string(min,max)	@string(7,10)	"0B(6!)5[)"
	@string(pool,min,max)	@string("lower",1,3)	"zz"
		@string("upper",1,3)	"OZJ"
		@string("number",1,3)	"61"
		@string("symbol",1,3)	"@%"
		@string("aeiou",1,3)	"au"
范围	@range(stop)	@range(10)	[0,1,2,3,4,5,6,7,8,9]

分类	规则	示例	示例结果
	@range(start,stop)	@range(3,7)	[3,4,5,6]
	@range(start,stop,step)	@range(1,10,3)	[1,4,7]

日期/时间

分类	规则	示例	示例结果
日期	@date	@date()	"2015-05-09"
	@date(format)	@date("yyyy-MM-dd")	"2012-11-08"
		@date("yy-MM-dd")	"10-06-12"
		@date("yyyy yy y MM M dd d")	"1971 71 71 05 5 02 2"
时间	@datetime	@datetime()	"1996-15-11 01:18:47"
	@datetime(format)	@datetime("yyyy-MM-dd A HH:mm:ss")	"1978-01-10 AM 03:59:54"
		@datetime("yy-MM-dd a HH:mm:ss")	"98-09-03 下午 19:32:44"
当前时间	@now	@now()	"2022-11-30 12:46:12"
	@now(unit)	@now("year")	"2022-01-01 00:00:00"
		@now("month")	"2022-08-01 00:00:00"
		@now("week")	"2022-08-09 00:00:00"
		@now("day")	"2022-08-11 00:00:00"
		@now("hour")	"2022-08-11 15:00:00"
		@now("minute")	"2022-08-11 15:24:00"
		@now("second")	"2022-08-11 15:24:02"

分类	规则	示例	示例结果
	@now(format)	@now("yyyy-MM-dd HH:mm:ss SS")	"2022-08-11 15:24:02 761"
	@now(unit,format)	@now("day", "yyyy-MM-dd HH:mm:ss SS")	"2022-08-11 00:00:00 000"
时间戳	@timestamp(format)	@timestamp("s")	"1662605353"
		@timestamp("ms")	"1662605408838"

Web 相关

规则	示例	示例结果
@url	@url()	"http://ihum.md/xmicdyn"
@url(protocol)	@url("http")	"http://ckpvjgyc.eg/jzatazzvm"
@url(protocol,host)	@url("http","test.com")	"http://test.com/pmuway"
@protocol	@protocol()	"ftp"
@domain	@domain()	"ljmhm.gu"
@domain(tld)	@domain("com")	"dtcboprfg.com"
@tld	@tld()	"com"
@email	@email()	"e.fsysqt@vjz.edu"
@email	@email("test.com")	"e.fsysqt@test.com"
@ip	@ip()	"xxx.xxx.xxx.xxx"

其他

分类	规则	示例	示例结果
GUID	@guid	@guid	"7C50585F-8DF1-4E94-B016-EE13BD36248D"
身份证ID	@id	@id	"712424201708225548"
uuid	@uuid	@uuid	"d29925a288404207b60161fdaed05d55"

Helper

分类	规则	示例	示例结果
全大写	@upper(str)	@upper("hello")	"HELLO"
全小写	@lower(str)	@lower("HELLO")	"hello"
多选一	@pick(arr)	@pick(["hello", "test", "test3"])	"hello"
		@pick([1, 5, 10, 60, 80])	10
		@pick([1,"hello","中国",true,80])	true

Name 相关

分类	规则	示例	示例结果
英文名	@first	@first()	"Michelle"
英文姓	@last	@last()	"Williams"
英文姓名	@name	@name()	"Michelle Williams"
	@name(middle)	@name(true)	"Karen Lisa Harris"
		@name(false)	"Karen Harris"
中文姓	@cfirst	@cfirst()	"张"
中文名	@clast()	@clast()	"三"
中文姓名	@cname	@cname()	"张三"

Address 相关

分类	规则	示例	示例结果
区域	@region	@region()	"西北"
省或直辖市、自治区、特别行政区	@province	@province()	"陕西省"
市	@city	@city()	"西安市"
	@city(prefix)	@city(true)	"陕西省 西安市"
		@city(false)	"西安市"
县	@county	@county()	"蓝田县"

分类	规则	示例	示例结果
	@county(prefix)	@county(true)	"陕西省 西安市 蓝田县"
		@county(false)	"蓝田县"
邮政编码	@zip	@zip()	"710500"

正则表达式

规则	示例	示例结果
@regexp(regexp)	@regexp("[0-9]+")	"36436"
	@regexp("[0-9]{3,5}")	"343"
	@regexp("[a-z]+\\@gmail\\.com")	"ifadt@gmail.com"

支持类型	描述
*	匹配前一个表达式0次或多次，等价于{0,}。
.	默认匹配除换行符之外的任何单个字符（用来生成乱码）。
+	匹配前面一个表达式1次或者多次，等价于{1,}。
?	匹配前面一个表达式0次或者1次，等价于{0,1}。
x|y	匹配'x'或者'y'。
{n}	n是一个正整数，匹配前面一个字符刚好出现了n次。
{n,}	n是一个正整数，匹配前面一个字符至少出现了n次。
{n,m}	n和m都是整数，匹配前面的字符至少n次，最多m次。如果n或者m的值是0，该值被忽略。
[xyz]	一个字符集合，匹配方括号中的任意字符。
@	与"."的功能类似，默认匹配除换行符之外的任何单个字符（生成乱码）。
\	将下一个字符标记为特殊字符、原义字符、向后引用或八进制转义符。例如： 'n'匹配字符'n'，'\n'匹配换行符，序列'\'匹配""，而'\('则匹配"("。
[	标记一个中括号表达式的开始。要匹配[，请使用[。
{	标记限定符表达式的开始。要匹配{，请使用{。

不支持类型	描述
^	匹配输入的开始。
$	匹配输入的结束。
(x)	匹配'x'并且记住匹配项，其中括号被称为捕获括号。
(?:x)	匹配'x'但是不记住匹配项，其中括号被称为非捕获括号，能够定义与正则表达式运算符一起使用的子表达式。
x(?=y)	匹配'x'仅仅当'x'后面跟着'y'，这种叫做先行断言。
(?<=y)x	匹配'x'仅当'x'前面是'y'，这种叫做后行断言。
x(?!y)	仅仅当'x'后面不跟着'y'时匹配'x'，这被称为正向否定查找。
(?<!y)x	仅仅当'x'前面不是'y'时匹配'x'，这被称为反向否定查找。
[^xyz]	一个反向字符集，即匹配任何没有包含在方括号中的字符。
[\b]	匹配一个退格(U+0008)。
\b	匹配一个词的边界。
\B	匹配一个非单词边界。
\cX	当X处于A到Z之间的字符时，匹配字符串中的一个控制符。
\d	匹配一个数字，等价于[0-9]。
\D	匹配一个非数字字符，等价于0-9。
\f	匹配一个换页符(U+000C)。
\n	匹配一个换行符(U+000A)。
\r	匹配一个回车符(U+000D)。
\s	匹配一个空白字符，包括空格、制表符、换页符和换行符。
\S	匹配一个非空白字符。
\t	匹配一个水平制表符(U+0009)。
\v	匹配一个垂直制表符(U+000B)。
\w	匹配一个单字字符（字母、数字或者下划线），等价于[A-Za-z0-9_]。
\W	匹配一个非单字字符，等价于[A-Za-z0-9_]。
\n	在正则表达式中，返回最后的第n个子捕获匹配的子字符串（捕获的数目以左括号计数）。
\0	匹配NULL（U+0000）字符。不能在后面跟其它小数，因为\0是一个八进制转义序列。
\xhh	匹配一个两位十六进制数（\x00-\xFF）表示的字符。
\uhhhh	匹配一个四位十六进制数表示的UTF-16代码单元。

不支持类型	描述
\u{hhhh} 或 \u{hhhhh}	仅当设置了u标志时，匹配一个十六进制数表示的Unicode字符。

说明

- “regexp”特殊字符需要加上“\”进行转义。
- 参考文档：**正则表达式**。

7 API 测试

7.1 API 测试导读

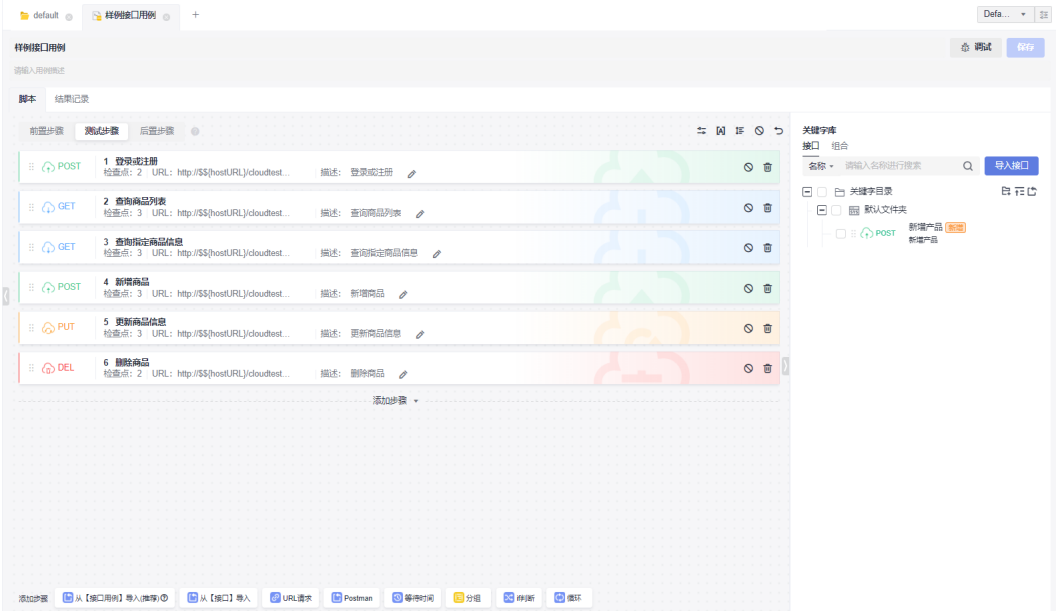
API 测试能力简介

CodeArts API的“API测试”功能来源于**CodeArts TestPlan**的“接口自动化测试”能力，且与CodeArts TestPlan的数据同源。

- CodeArts API针对API做测试，更贴近API的设计、开发到测试的同步场景。
- CodeArts TestPlan对整个项目做整体的测试计划与测试方案，更偏向测试场景使用。

接口自动化测试

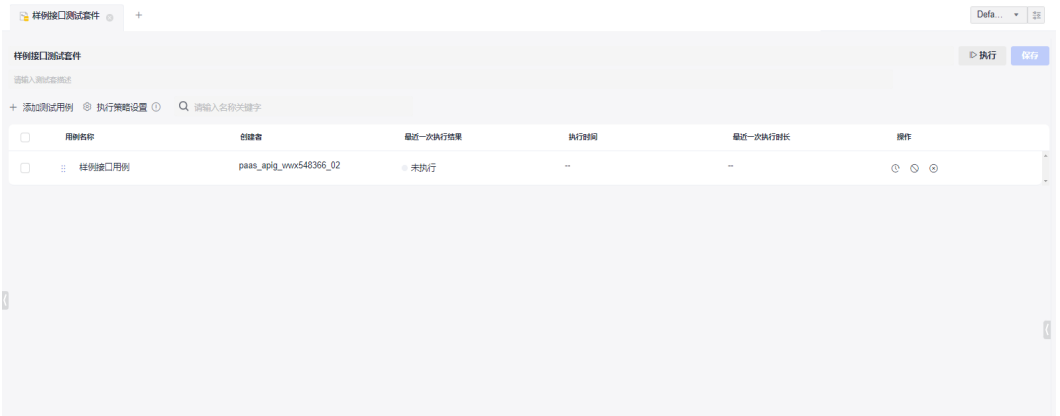
接口自动化测试提供了丰富的用户可视化编辑页面，支持测试步骤编排，以及测试步骤的检查点和响应结果参数提取，帮助用户快速生成与API设计相匹配的接口测试用例，可实现“零代码”快速生成测试用例，将测试人员从复杂的工作中释放出来，提升软件接口的测试效率。



自动化测试套件

CodeArts API支持创建接口自动化测试套件，用户可以将多个测试用例组合成为一个测试套件，测试套件的功能具有以下特点：

- 测试套件将一组测试用例编排组合，实现更复杂的集成测试场景。
- 测试套件支持并行/串行执行。
- 测试套件支持配置多种执行策略，可以配置定时执行、执行周期、执行时间等。



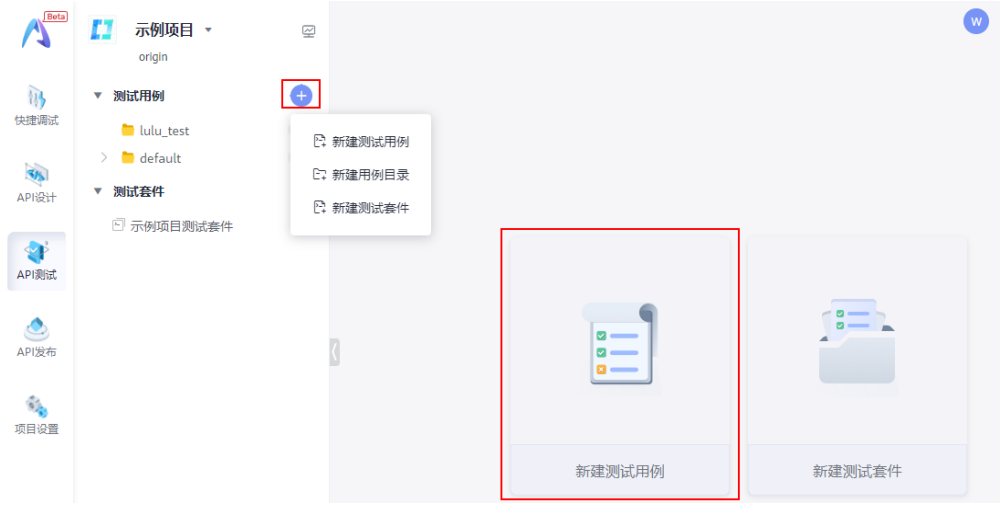
7.2 测试用例管理

7.2.1 新建测试用例

操作步骤

1. 单击“API测试”菜单，进入测试用例管理主页面，可通过以下两种入口新建测试用例。

- 入口一：单击主页面的“新建测试用例”选项。
- 入口二：单击左侧测试用例旁边的 ，选择下拉选项中的“新建测试用例”选项。



2. 进入“新建测试用例”页面，输入测试用例的名称，根据需要配置名称（必填）、描述、前置步骤、测试步骤、后置步骤等信息，单击“保存”，即可完成用例创建。

7.2.2 添加测试步骤

接口自动化测试用例可包括三个阶段：准备阶段、测试阶段、销毁阶段。

准备阶段对应的“前置步骤”，即测试的前置准备条件；测试阶段对应“测试步骤”，实现接口的功能测试；销毁阶段对应“后置步骤”，实现准备阶段和测试阶段数据的释放或恢复，保证测试用例可重复执行。

- 准备阶段（可选）：前置步骤。
  - 在此阶段中准备测试阶段需要的前置条件数据，如果没有前置条件，可以忽略此阶段。

  - 如果前置条件的数据需要在测试阶段中引用，可以使用参数传递将数据参数化后，传递给测试阶段引用，详见**响应提取**。
- 测试阶段：测试步骤。

  定义接口核心测试步骤，主要是对测试场景的详细描述，建议从API设计中选择要测试接口作为核心测试步骤，在此步骤检测最终结果。
- 销毁阶段（可选）：后置步骤。

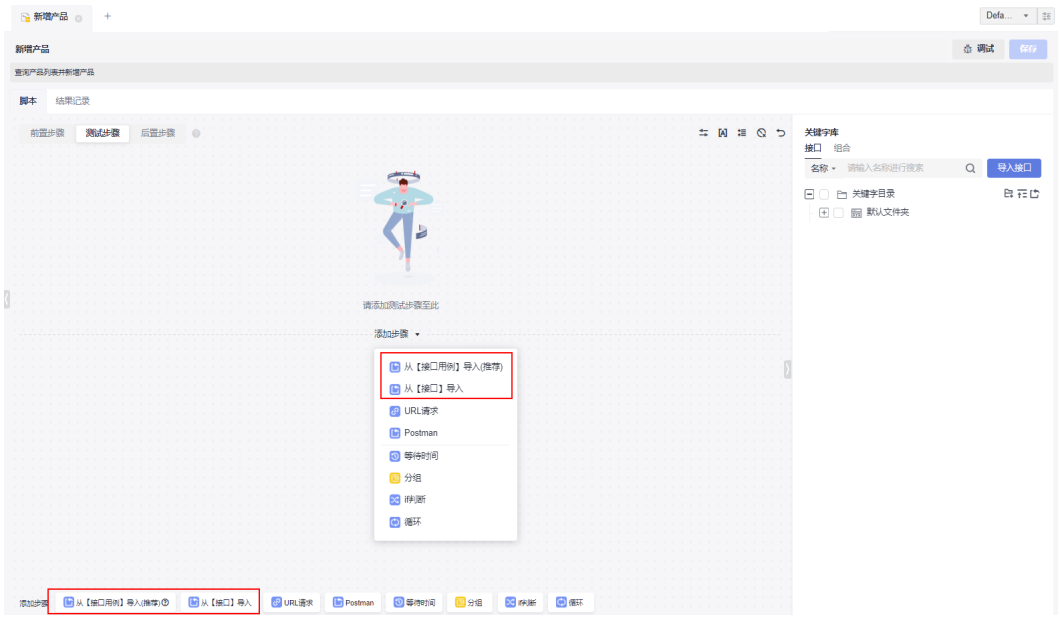
  为了不影响其它测试用例执行或保证当前测试用例可重复执行，建议在每次测试结束后还原测试数据。如果没有数据需要销毁，可以忽略此阶段。

操作步骤

创建接口自动化测试用例后，在测试用例页面，单击“脚本 > 测试步骤”页签，可以使用多种方式添加测试步骤。

从接口/接口用例导入

用户可以将API设计的数据导入测试步骤，主要有两个来源：“从【接口用例】导入（推荐）”和“从【接口】导入”。

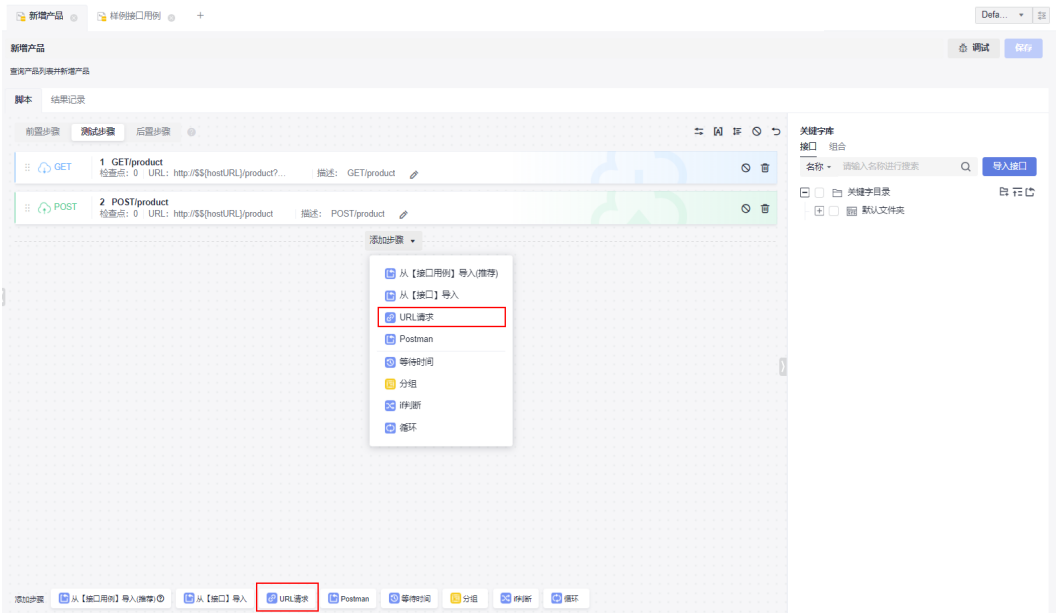


- 从【接口用例】导入（推荐）
  a. 在弹出的“从【接口用例】导入（推荐）”对话框上，选择需要导入的接口，即“API设计”中接口的调试用例，参考**保存为用例**。
  b. 单击“导入”，完成测试步骤添加。

- 从【接口】导入
  a. 在弹出的“从【接口】导入”对话框上，选择需要导入的接口，即“API设计”的接口数据。
  b. 单击“导入”，完成测试步骤添加。

自定义请求

在测试用例中，可能需要调用项目之外的接口，例如在请求华为终端app前，需要调用账号服务的获取用户认证token。此时用户可以在测试步骤中添加自定义请求，自定义请求可以是任何 HTTP 请求，包括常见的 GET、POST、PUT、DELETE 等请求。



1. 单击“URL请求”选项，添加“自定义URL请求”测试步骤。
2. 在自定义请求中可编辑请求名称、地址和请求参数等信息，详细参照**设置接口请求**章节。

从 postman 文件导入

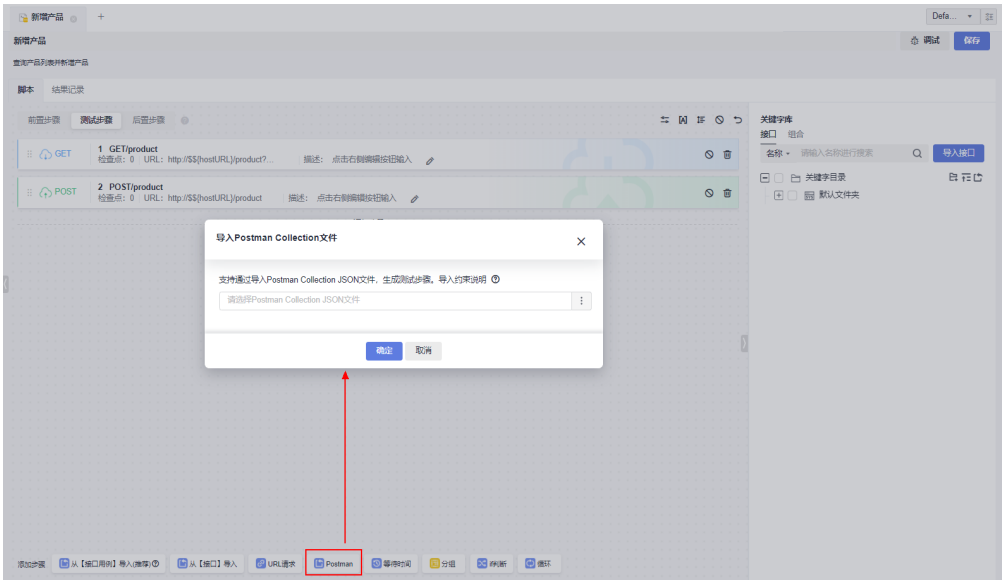
接口自动化测试用例支持通过导入Postman文件，生成测试步骤。

导入Postman文件需满足以下要求：

- 支持Postman Collection v2.1标准。
- 仅支持Postman请求方法、请求URL、请求头、请求体生成测试步骤。
- Postman请求体导入方式仅支持form-data、x-www-form-urlencode、raw。

- Postman请求体form-data上传附件需要在测试步骤中单独上传。

1. 单击“Postman”，弹出“导入Postman Colection文件”对话框。

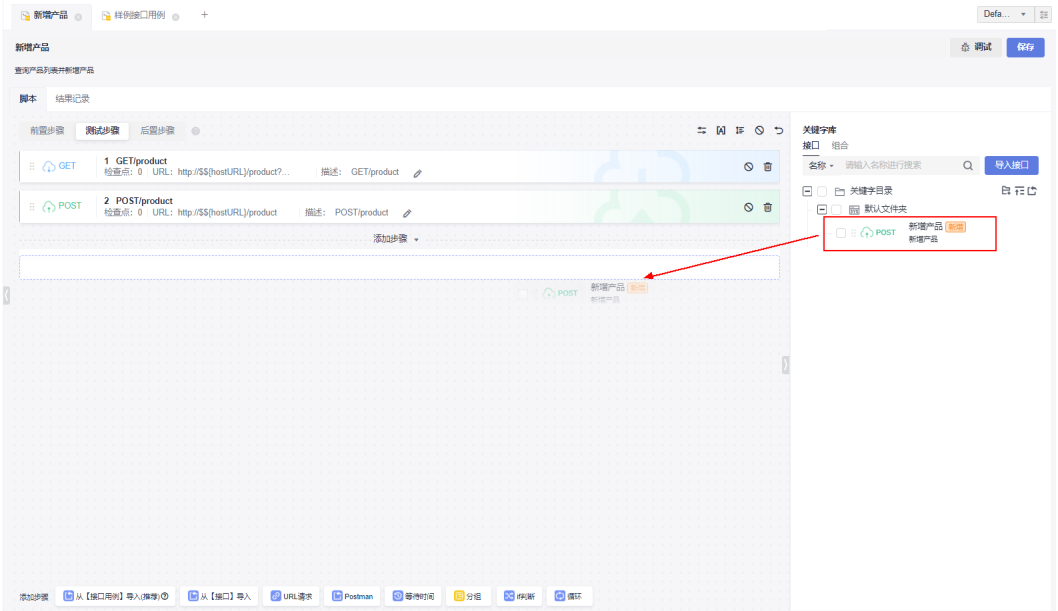


2. 上传需要导入的postman文件,单击“确定”，导入postman请求文件，完成测试步骤添加。

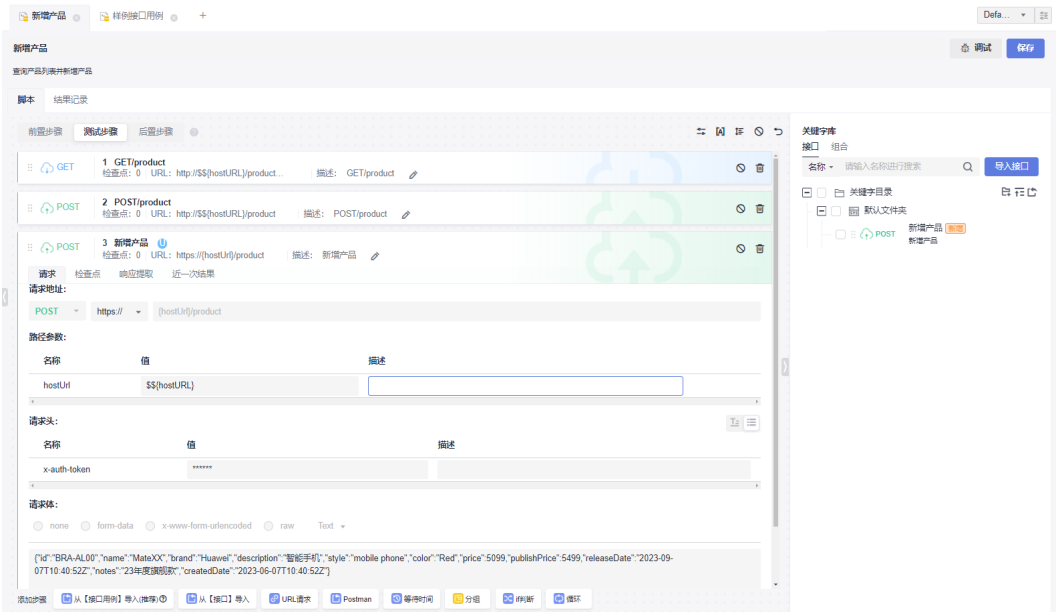
关键字导入

支持从关键字库中添加测试用例步骤，如何添加及使用关键字库请参加7.2.3 关键字库。

可以采用拖拽的方式，将关键字设置为测试步骤。



添加后，支持对测试步骤修改，且不会影响关键字库中的数据。



7.2.3 关键字库

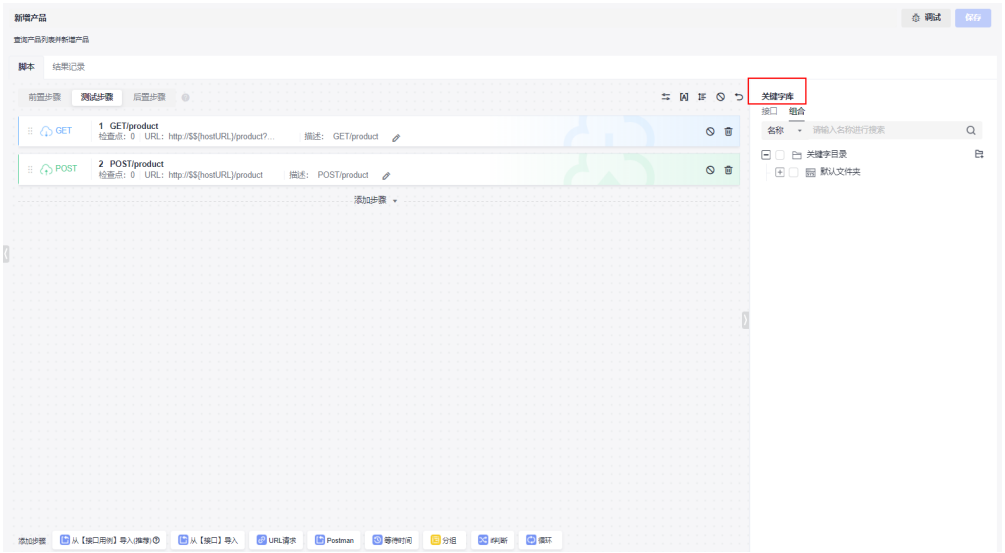
关键字驱动测试是一种测试自动化的技术，通过提供一组称为关键字的“构建块”创建自动化测试用例。关键字驱动测试可用于组件测试、系统测试等不同级别的测试，其优势体现在易用性、可理解性、可维护性、测试信息的重用、支持测试自动化、节约潜在的成本和进度等方面。在设计测试用例时，经常会遇到一些相同的前置步骤或者测试逻辑。如果每一个测试用例中都编写这些步骤，重复工作量很大，并且难以维护。测试关键字可以帮助复用这些测试步骤。

关键字库将接口关键字、组合关键字、系统关键字进行统一管理，打造一站式关键字管理能力，在设计用例脚本时用户体验保持一致。

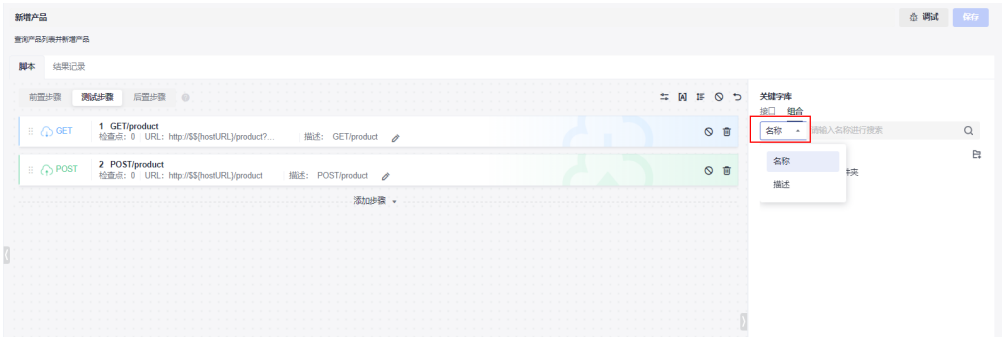
- 接口关键字中定义单个接口的请求，可以通过导入Swagger文件、保存自定义URL请求等方式生成。
- 组合关键字用于将多个步骤封装成常用的测试步骤组合，可以引入到其他的测试用例中，实现测试逻辑复用。
- 系统关键字涵盖认证、协议、中间件、数据库四大类别，覆盖身份认证、复杂协议、数据处理、数据预置、数据验证、接口集成等丰富场景。

关键字库入口

1. 在“脚本”页签，页面右侧展示“关键字库”。



   - 通过Swagger文件导入的关键字放置在关键字库的“接口”页签中，详细操作请参考**接口关键字**。
   - 关键字用例和组合关键字放置在关键字库的“组合”页签中，支持从0到1创建新的组合关键字，详细操作请参考**组合关键字**。
   - 认证类、数据库操作类、中间件类、协议类等实用关键字放置在“系统”页签中，详细操作请参考**接口自动化用例关键字**。
2. 单击关键字库中“名称”下拉列表，可以按照“名称”、“描述”，搜索对应的关键字。

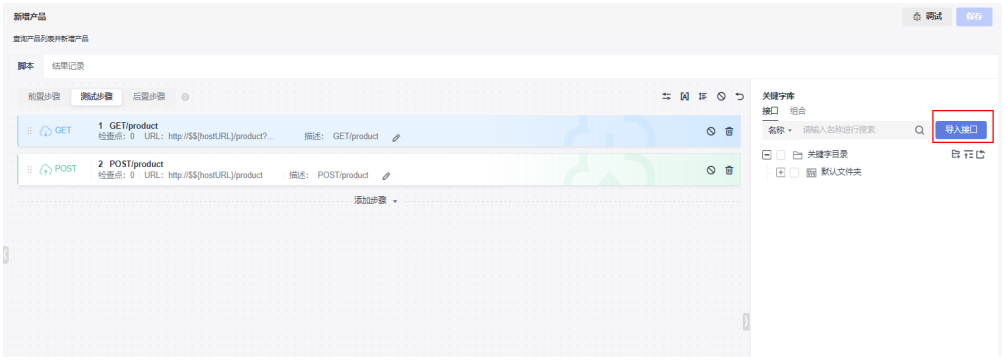


接口关键字

接口关键字中定义单个接口的请求，可以通过导入Swagger文件、保存自定义URL请求等方式生成。

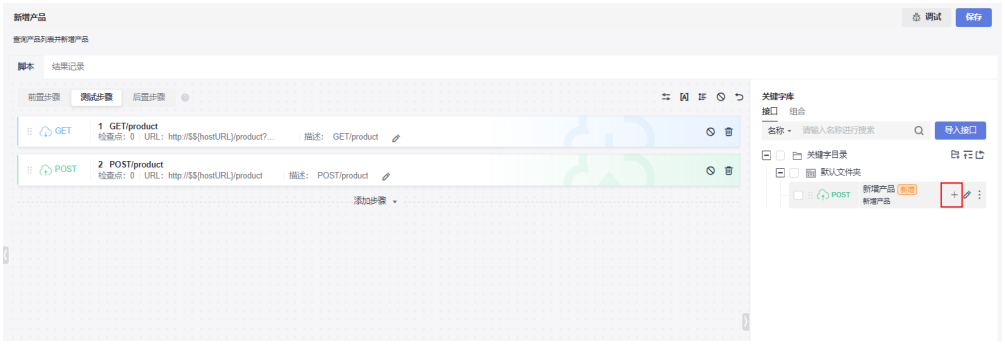
导入Swagger文件生成数据

1. 在右侧“关键字库”界面，单击“导入接口”。



2. 在弹出的“导入接口”页面，单击“点击添加文件或拖拽上传”，选择配置好的Swagger接口文档，单击“确定”，完成Swagger文件导入。

3. 文档导入成功后，系统自动解析生成脚本模板，脚本模板包含了接口的基本描述信息。可以从“关键字库”界面单击“+”或直接拖拽接口添加至测试步骤，只需要按模板输入接口参数就可以进行测试。



具体请参考CodeArts Testplan的**接口关键字**。

组合关键字

在设计测试用例时，经常会遇到一些相同的前置步骤或者测试逻辑。如果每一个测试用例中都编写这些步骤，重复工作量很大，并且难以维护。组合关键字用于将多个步骤封装成常用测试逻辑，可以被测试用例调用实现逻辑复用。

说明

使用组合关键字高阶特性，需要特性使用授权，请联系客服处理。

具体请参考CodeArts Testplan的**组合关键字**。

接口自动化用例系统关键字

接口自动化测试将常见的操作封装成关键字供用户使用，用于提升编写接口测试用例的效率。

说明

使用系统关键字高阶特性，需要开通TestPlan专业版套餐或CodeArts专业版以上套餐。

表 7-1 系统关键字一览表

分类	关键字集
认证类	• 认证类—GetIAMToken • 认证类—OAuth2ForOneAccess

分类	关键字集
数据库操作类	• 数据库类—MySQLQuery • 数据库类—MySQLUpdate • 数据库类—MySQLInsert • 数据库类—MySQLDelete • 数据库类—OpenGaussQuery • 数据库类—OpenGaussUpdate • 数据库类—OpenGaussInsert • 数据库类—OpenGaussDelete • 数据库类—PostgreSQLQuery • 数据库类—PostgreSQLUpdate • 数据库类—PostgreSQLInsert • 数据库类—PostgreSQLDelete • 数据库类—MongoDBQuery • 数据库类—MongoDBInsert • 数据库类—MongoDBUpdate • 数据库类—MongoDBDelete
中间件类	• 中间件类—RedisGet • 中间件类—RedisSet • 中间件类—OBSWrite • 中间件类—OBSDelete • 中间件类—OBSQuery • 中间件类—KafkaProducer • 中间件类—KafkaConsumer
协议类	• 协议类—TCP • 协议类—UDP • 协议类—WSConnect • 协议类—WSRequest • 协议类—WSWriteOnly • 协议类—WSReadOnly • 协议类—WSDisConnect • 协议类—MQTTConnect • 协议类—MQTTDisconnect • 协议类—MQTTPublish • 协议类—MQTTSubscribe • 协议类—DubboClient

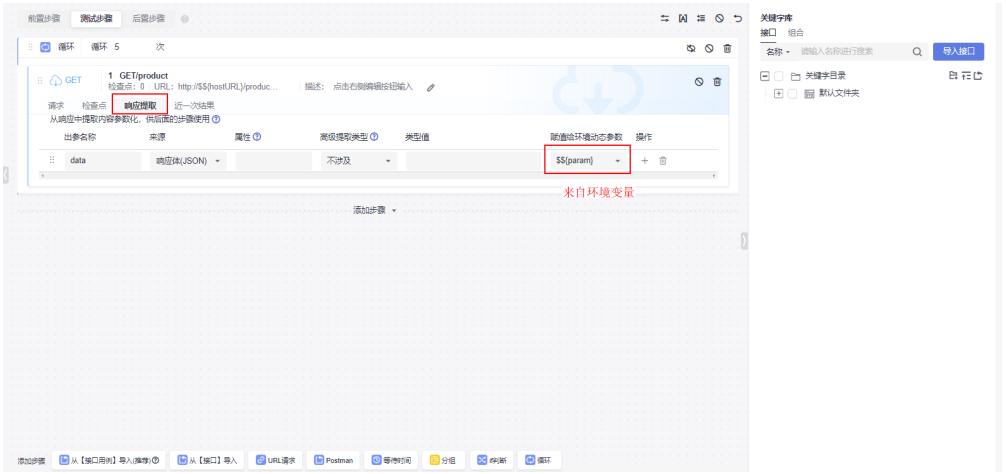
具体请参考CodeArts Testplan的**接口自动化用例系统关键字**。

7.2.4 响应提取

响应提取是提取接口响应结果的某一部分，命名为参数，供后续测试步骤参数化调用。响应提取需要在前序测试步骤定义，后续测试步骤使用。

- 在前序测试步骤中，在“响应提取”页签创建要传递的参数。响应提取来源用到内置参数，请参考**内置参数**了解如何使用内置参数。响应提取同时支持正则表达式的匹配，提取出与给定正则表达式匹配的返回值。
- 在后续测试步骤中，通过“${参数名}”方式引用前序测试步骤创建的响应提取。后续步骤的URL、请求头、请求体中均可以引用此参数。如果在JSON格式的请求体中引用此参数，请在参数外使用英文引号，如：

```
{
id: "用例ID"
name:"${name}"
}
```



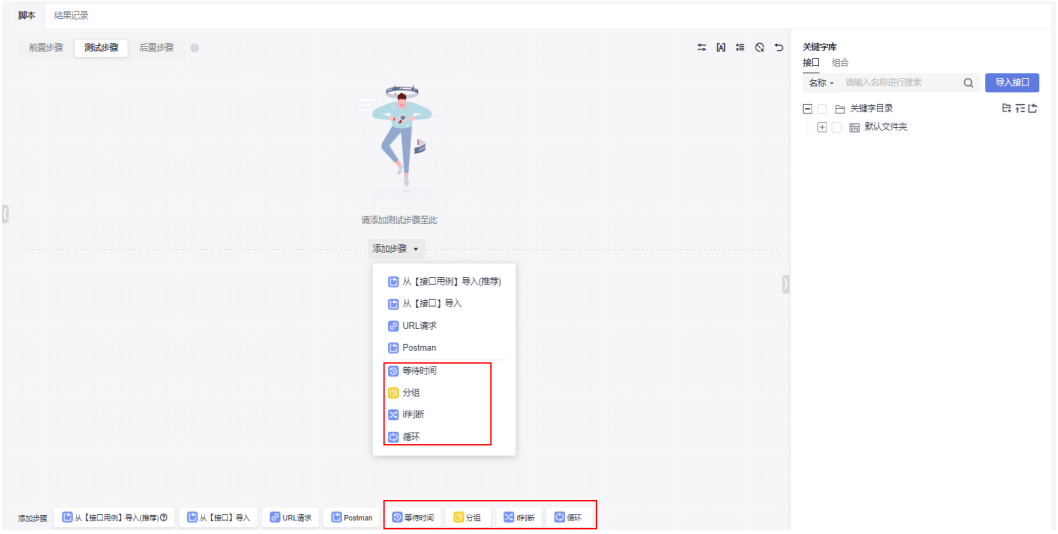
- 响应提取支持根据给定的“key:value”获取字符串，详细配置可参考**示例：根据给定的key:value从响应体中获取字符串**。

字段	说明
出参名称	用于之后使用${出参名称}来引用此参数，名称使用字母数字下划线。
来源	被检测字段的来源，如响应体（JSON）、响应头、响应码。

字段	说明
属性	• 若来源是响应码，属性为空。详细介绍请参见**响应码检查**。 • 若来源是响应头，属性为响应头中字段的名称。详细介绍请参见**响应头检查**。 • 若来源是响应体（JSON），属性有两种填写方法： 1. 普通提取表达式（非“$”开头），例如“item.name”。取字段中的值，支持嵌套取值。详细介绍请参见**响应体（JSON）检查**。 从响应体中提取数组时，下标可以是数字，也可以是“key:value”表达式，详细介绍请参见**示例：根据给定的key:value从响应体中获取字符串**。 2. JsonPath表达式(“$.”或“$[”开头)，例如“$.store.book[0].title”。 详细介绍请参见**示例：根据JsonPath从响应体中获取数据**。
高级提取类型	可选项，使用高级提取类型，辅助提取响应结果信息，若选择不涉及则视为不使用额外的方式匹配。 目前有两种方式： • **字符串提取**，也就是字符串的截取。 • **正则表达式**，即使用正则方式对来源字符串进行过滤。 高级提取类型优先使用字符串提取功能，若不能满足需求可考虑使用正则表达式。
赋值给环境动态参数	将响应提取后的值赋值给动态变量，用于后续测试引用。请参考**新建变量**，如何设置动态变量。

7.2.5 测试流程控制条件

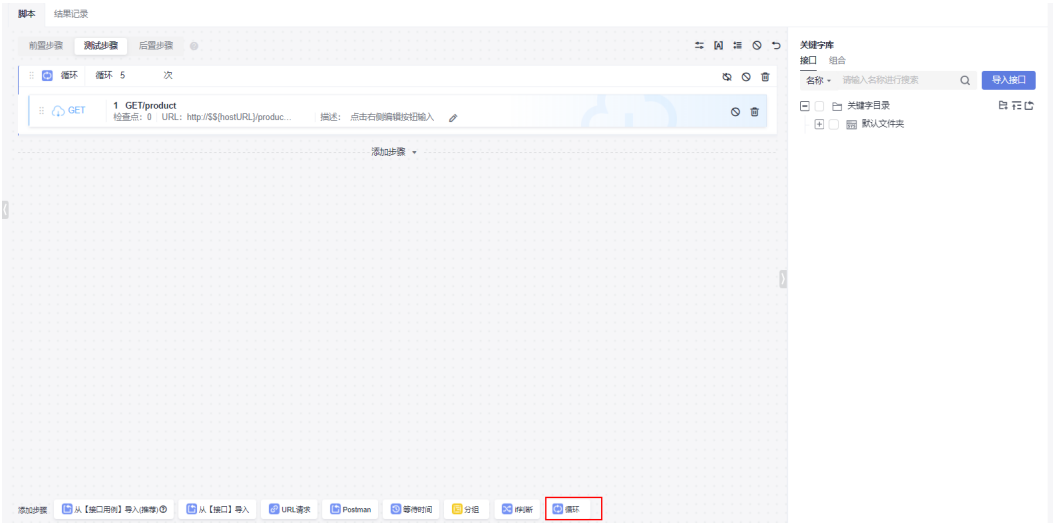
在测试用例中，可能会有复杂的测试场景，因此CodeArts API提供了多种流程控制条件：循环、判断、分组、等待时间。



循环

当测试步骤需要重复执行时，可以指定重复执行次数。

1. 进入接口自动化用例编辑的“脚本”页签，单击“循环”。
2. 设置循环次数。
3. 循环中可添加URL请求、判断、分组、等待、测试关键字。

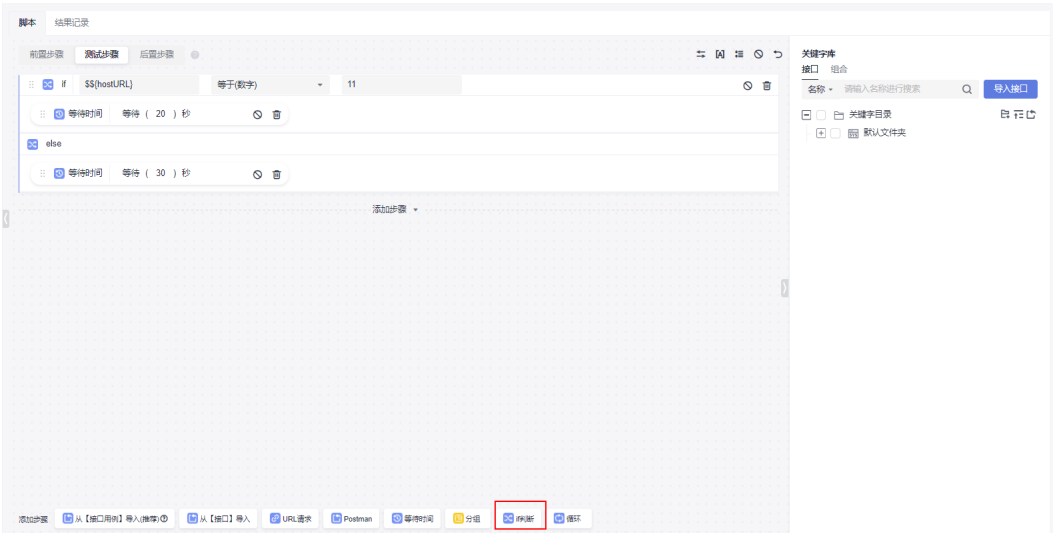


判断

如果需要根据前序测试步骤的结果决定后续需要执行的测试步骤，可使用“if判断”。

判断设置方式如下：

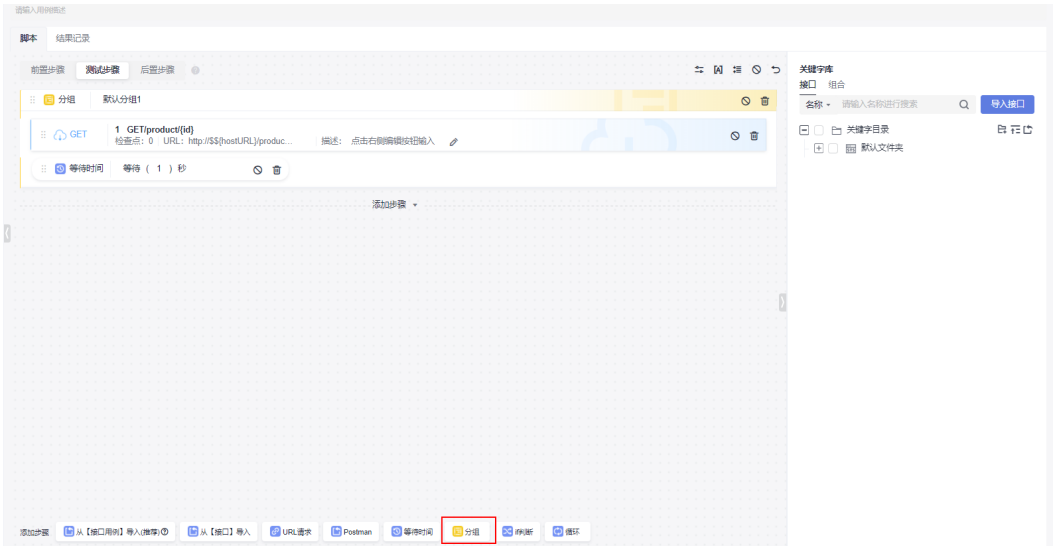
1. 进入接口自动化用例编辑的“脚本”页签，单击“if判断”。
2. 添加判断逻辑，输入if判断条件，在分支中添加后续测试步骤。
3. 判断分支中可添加URL请求、判断、等待、循环、测试关键字。



分组

当测试流程中多个步骤存在关联关系时，可以对测试步骤进行归类，并放到同一分组中。通过对测试步骤的分组，让测试场景具备更好的可读性和操作性。

1. 进入接口自动化用例编辑的“脚本”页签，单击“分组”。
2. 输入分组名称，并拖拽相关的测试步骤到分组。
   - 分组中可添加URL请求、判断、等待、循环。
   - 支持拖拽编排分组在测试用例中的顺序。
   - 支持拖拽编排分组内部测试步骤的顺序。
   - 支持整体禁用或者删除分组。

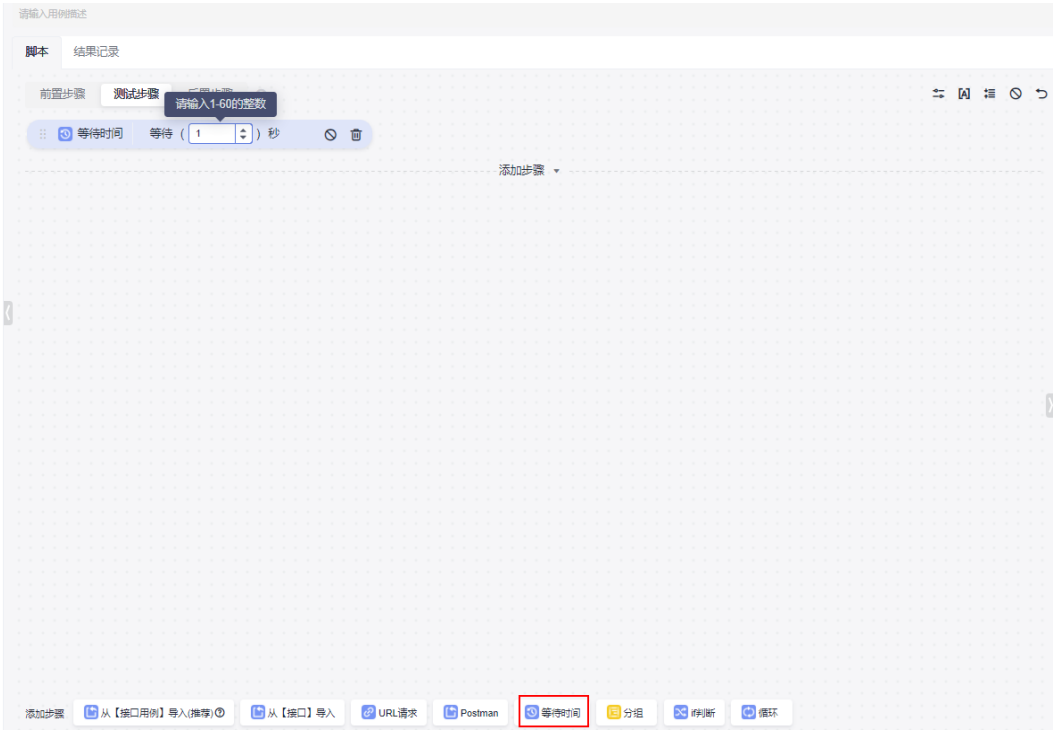


等待时间

如果执行某个测试步骤后需要间隔一段时间再继续执行后续测试步骤，可以使用等待时间。

等待时间设置方式如下：

1. 进入接口自动化用例编辑的“脚本”页签，单击“等待时间”。
2. 输入等待时间，支持范围在1~60的整数。



7.2.6 测试检查点

在测试步骤请求中可以设置请求结果的检查点，方便用户检测当前接口的返回值，验证测试步骤是否得到预期结果。

设置建议

- 推荐用户设置检查点。针对接口类请求，请提供判断响应码的检查点。
- 当检查点设置为空，不论接口的响应码是什么，都会判断结果为成功。

检查点说明

测试检查点即请求结果的断言，通过检查接口响应是否符合预期，判定系统是否满足预期。

在接口自动化用例中，测试步骤的“检查点”页签中可以定义测试检查点。

检查点内容包括属性、比较符和目标值定义。

字段	说明
重试次数	如果检查点失败，重新执行该测试步骤的次数，重试次数范围：0~5的正整数。
重试间隔	如果检查点失败，每次重试的间隔时长，单位为ms，重试间隔范围：0~10,000的正整数。
来源	被检测字段的来源，如响应体（JSON）、响应头、响应码、变量。
属性	• 若来源是响应码，属性可为空。详细介绍请参见**响应码检查**。 • 若来源是响应头，属性为响应头中字段的名称。详细介绍请参见**响应头检查**。 • 若来源是响应体（JSON），属性有两种填写方法： – 普通提取表达式（非“$”开头），例如“item.name”。取字段中的值，支持嵌套取值。详细介绍请参见**响应体（JSON）检查**。 从响应体中提取数组时，下标可以是数字，也可以是“key:value”表达式，详细介绍请参见**示例：根据给定的key:value从响应体中获取字符串**。 – JsonPath表达式(“$.”或“$[”开头)，例如“$.store.book[0].title”。 详细介绍请参见**示例：根据JsonPath从响应体中获取数据**。 • 若来源是变量，属性为全局变量，局部变量，响应提取后的变量。详细介绍请参见**变量检查**。
高级提取类型	可选项，使用高级提取类型，辅助提取检查点信息，若选择不涉及，则视为不使用额外的方式匹配。 目前有两种方式： • **字符串提取**，也就是字符串的截取。 • **正则表达式**，即使用正则方式对来源字符串进行过滤。 高级提取类型优先使用字符串提取功能，若不能满足需求可考虑使用正则表达式。
类型值	高级提取类型中所需的参数。
比较符	支持数字、字符串、JSON对象比较、类型比较等多种比较符，详细介绍请参见**比较符说明**。
目标值	检查点期望值。目标值支持使用内置参数，请参考**内置参数**了解如何使用内置参数。

例如，检查响应体（JSON格式）中“item.name”字段的第零位（首位）到第四位之间是不是petty，参数配置如下：

字段	值
来源	响应体（JSON）

字段	值
属性	item.name
高级提取类型	字符串提取
类型值	0、5
比较符	等于(字符串)
目标值	petty

比较符说明

支持以下比较类型：

比较类型	比较符	是否需要填入值	举例
数字比较	• 等于(数字) • 不等于(数字) • 大于等于(数字) • 小于等于(数字) • 大于(数字) • 小于(数字)	是	• 响应码 等于 200 • 响应码 不等于 200 • 响应码 大于等于 200 • 响应码 小于等于 200 • 响应码 大于 200 • 响应码 小于 200
字符串比较	• 等于(字符串、区分大小写) • 不等于(字符串、区分大小写) • 等于(字符串、不区分大小写) • 包含(字符串) • 不包含(字符串)	是	• 响应体中的属性param1 等于 test • 响应体中的属性param2 不等于 test • 响应体中的属性param3 等于 TEST • 响应体中的属性param4包含 tri • 响应体中的属性param5不包含 tri
正则比较	正则表达式	是	• 响应体中的属性param1 正则表达式 ^[A-Za-z0-9]{1,32}$
通用比较	• 是空（该字段是否为空，即是否没有该字段） • 不是空（该字段是否不为空，即是否有该字段）	否	• 响应体中的属性param1 是空 • 响应体中的属性param2 不是空

比较类型	比较符	是否需要填入值	举例
Json数组比较	• JSON数组为空数组（json数组是否为空数组） • JSON数组为非空数组（json数组是否为非空数组）	否	• 响应体中的属性param1 JSON数组为空数组 • 响应体中的属性param2 JSON数组为非空数组
	JSON数组的大小（json数组的大小是否为...）	是	• 响应体中的属性param1 JSON数组的大小
类型比较	• 是JSON类型（是否是json对象类型） • 是JSON数组类型（是否是json数组类型）	否	• 响应体中的属性param1 是JSON类型 • 响应体中的属性param2 是JSON数组类型
Json对象比较	JSON等于（json等于）	是	• 响应体中的属性param1 JSON等于 {"name":"zhangsan"}

响应码检查

将响应码与目标值进行比较，例如：检查响应码是否等于“200”。

响应头检查

响应头中字段的值和目标值进行比较，例如：检查响应头中“content-type”的值是否等于“text/plain;charset=UTF-8”。

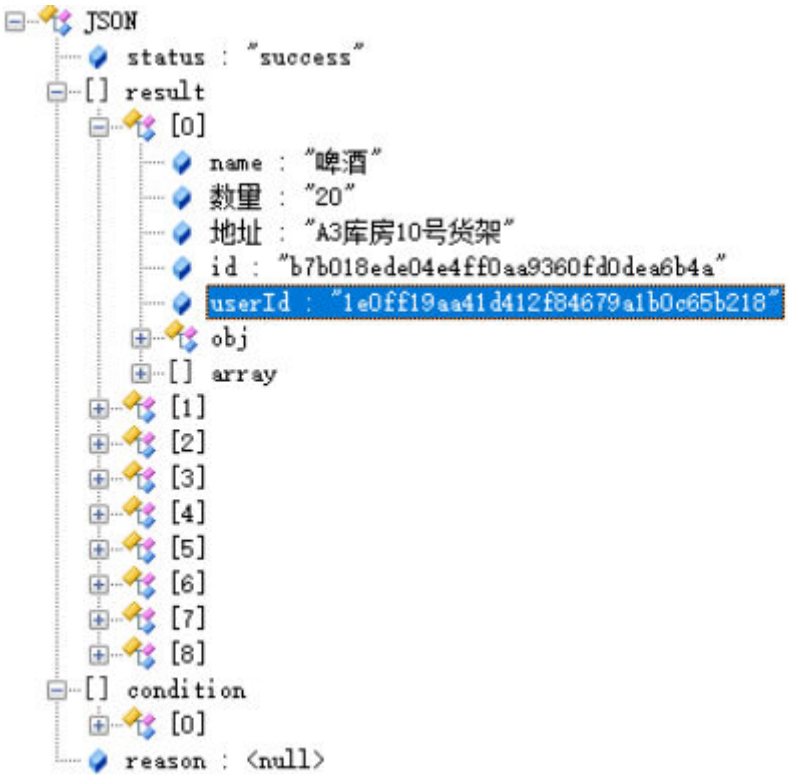
响应体（JSON）检查

1. 检查响应体（JSON）中对象字段的值。例如：
   响应体结构如下：

检查响应体对象中字段名为“status”的值，检查点配置如下：

2. 检查响应体（JSON）中某个数组的对象的字段值（数组条件采用下标确定对象，下标从0开始）。例如：

   响应体结构如下：



   检查响应体“result”数组的第1个元素对象字段名为“userId”的值，检查点配置如下：

3. 检查响应体（JSON）中某个数组的对象的字段值（数组条件采用模糊匹配功能确定对象）。例如：

   响应体结构如下：

   a.检查响应体“result”数组下“name”是“啤酒”的所有对象取第1个之后获取id的值，检查点配置如下：

说明

当获取的数组下的对象只有一个，“[0]”可以省略，示例中的表达式可以写为“result[name==啤酒].id”。

b.检查响应体“result”数组下“name”是“啤酒”的并且“obj”对象下属性“a”的值是“2”的对象的id值，检查点配置如下：

变量检查

检查比较全局变量，局部变量，响应提取后的变量值和目标值。例如：

- 检查全局变量“hostIp”的值是否等于“127.0.0.1”。
- 检查局部变量“local”的值是否等于“text”。
- 检查响应提取后的变量“name”的值是否等于“白酒”。

示例：根据给定的 key:value 从响应体中获取字符串

检查体的“属性”字段，可输入下标为“key:value”格式的数组，用以实现根据给定的“key:value”从响应体中获取满足该条件的json串。

“key:value”之间支持以下连接符：

连接符	说明	举例
==	等于（字符串、数值），支持汉字。	name==张三 age==20
!=	不等于（字符串、数值），支持汉字。	name!=张三 age!=20
>	大于（数值）	age>20
>=	大于等于（数值）	age>=20
<	小于（数值）	age<20
<=	小于等于（数值）	age<=20

数组下标支持以“&&”连接的多组“key:value”条件，表示提取的JSON串需要同时满足所有的“key:value”条件。例如：student[age>20&&gender=女]，表示提取条件为年龄大于20岁且性别为女。

数组的搜索条件中支持嵌套数组，在此情况下子嵌套条件中的对象必须全部满足条件，才能使得子条件成立。

说明

- 若value值为空对象，则需要使用$null，即key==$null。
- 若value值是null串（"null"），则使用null，即key==null。
- 若value值是空串（""），则可以不写，即key==。

以下面的响应体（JSON格式）为例：

```
{
  "status": "success",
  "result": [
    {
      "name": "啤酒",
      "数量": "20",
      "地址": "A3库房10号货架",
      "obj": {
        "a": 1,
        "b": "test",
        "c": "测试"
      },
      "array": [
        {
          "id": 1,
          "name": "aaa"
        },
        {
          "id": 2,
          "name": "bbb"
        }
      ]
    },
    {
      "name": "啤酒",
      "数量": "10",
      "地址": "A3库房10号货架",
      "obj": {
        "a": 1,
        "b": "test",
        "c": "测试"
      },
      "array": [
        {
          "id": 1,
          "name": "aaa"
        },
        {
          "id": 2,
          "name": "bbb"
        }
      ]
    },
    {
      "name": "白酒",
      "数量": "20",
      "地址": "A3库房10号货架",
      "obj": {
        "a": 1,
        "b": "test",
        "c": "测试"
      },
```



```
    "array": [
      {
        "id": 1,
        "name": "aaa"
      },
      {
        "id": 2,
        "name": "bbb"
      }
    ]
  },
  {
    "name": "白酒",
    "数量": "30",
    "地址": "A3库房10号货架",
    "obj": {
      "a": 1,
      "b": "test",
      "c": "测试"
    },
    "array": [
      {
        "id": 3,
        "name": "aaa"
      },
      {
        "id": 4,
        "name": "bbb"
      }
    ]
  },
  {
    "name": null,
    "数量": "10",
    "地址": "A3库房10号货架",
    "obj": {
      "a": 2,
      "b": "test",
      "c": "测试"
    },
    "array": [
      {
        "id": 5,
        "name": "aaa"
      },
      {
        "id": 6,
        "name": "bbb"
      }
    ]
  },
  {
    "name": "",
    "数量": "10",
    "地址": "A3库房10号货架",
    "obj": {
      "a": 2,
      "b": "test",
      "c": "测试"
    },
    "array": [
      {
        "id": 5,
        "name": "aaa"
      },
      {
        "id": 6,
        "name": "bbb"
      }
```

```
    ]
   }
 ],
 "condition": [
   {
     "name": "啤酒",
     "数量": "120",
     "地址": "A1库房15号货架"
   }
 ],
 "reason": null
}
```

从响应体中获取数据与对应的表达式如下表所示：

提取条件	表达式
获取result数组中name是白酒的数据	result[name==白酒]
获取result数组中name不是啤酒且数量大于20的数据	result[name!=啤酒&&数量>20]
获取result数组中name是空对象且array数组中id小于等于2的数据	result[name==$null&&array[id<=2]]
获取result数组中name是null串且obj对象的属性a等于2的数据	result[name==null&&obj.a==2]
获取result数组中name是空串（""）的数据	result[name==]

若需要检查（JSON格式）name是白酒、且数量大于20的数据是不是“[{"name": "白酒","数量": "30","地址": "A3库房10号货架","obj": {"a": 1,"b": "test","c": "测试"},"array": [{"id": 3,"name": "aaa"},{"id": 4,"name": "bbb"}]}]”，则检查点各字段的配置可参考下表：

字段	值
来源	响应体（JSON）
属性	result[name!=啤酒&&数量>20]
比较符	等于(字符串)
目标值	[{"name": "白酒","数量": "30","地址": "A3库房10号货架","obj": {"a": 1,"b": "test","c": "测试"},"array": [{"id": 3,"name": "aaa"},{"id": 4,"name": "bbb"}]}]

示例：根据 JsonPath 从响应体中获取数据

JsonPath详情可参见**官网**。

以下面的响应体（JSON格式）为例：

```
{
   "store": {
```



```
    "book": [
      {
        "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95
      },
      {
        "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99
      },
      {
        "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99
      },
      {
        "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95
    }
  },
  "expensive": 10
}
```

JsonPath表达式可以使用点符号或者括号符号，例如：

- $.store.book[0].title
- $['store']['book'][0]['title']

JsonPath表达式与其对应的含义如下表所示：

JsonPath表达式	含义描述
$.store.book[*].author	所有书籍的作者
$..author	所有作者
$.store.*	所有的东西，包括书籍和自行车
$.store..price	所有东西的价格
$..book[2]	第三本书
$..book[-2]	倒数第二本书
$..book[0,1]	前两本书
$..book[:2]	从索引 0（包括）到索引 2（不包括）的所有书籍
$..book[1:2]	从索引 1（包括）到索引 2（不包括）的所有书籍

JsonPath表达式	含义描述
$..book[-2:]	最后两本书
$..book[2:]	从末尾开始的两本书
$..book[?(@.isbn)]	所有有isbn的书
$.store.book[?(@.price < 10)]	商店里所有价格小于10的书
$..book[?(@.price <= $['expensive'])]	商店里所有不“贵”的书
$..book[?(@.author =~ /.*REES/i)]	所有作者匹配正则表达式.*REES（忽略大小写）的书籍
$..*	所有内容

说明

若使用length函数或者size函数，则多次调用时不要使用深度扫描（即“..”符号），例如“$..book.length()”返回书的数量，此时需要确定路径，改为“$.store.book.length()”。

7.2.7 环境&参数

在自动化测试中，通常存在多个测试环境，每个测试环境的环境参数值会有不同，例如域名、账号等。这些参数常会在测试脚本中被使用到，如果将这些参数硬绑定到测试脚本中，脚本的冗余度很高，复用度很低。

合理的测试设计要求分离测试逻辑和测试数据，实现测试逻辑的最大化复用，增强测试用例的可维护性和投入产出比。例如不同测试环境的URL域名是一种独立于测试逻辑，和测试环境相关的测试数据。测试用例参数可以用来管理这些测试数据。

接口自动化中的测试用例参数分为三类：

- **全局参数**
- **局部参数**
- **内置参数**

参数优先级：内置参数 > 局部参数 > 全局参数。

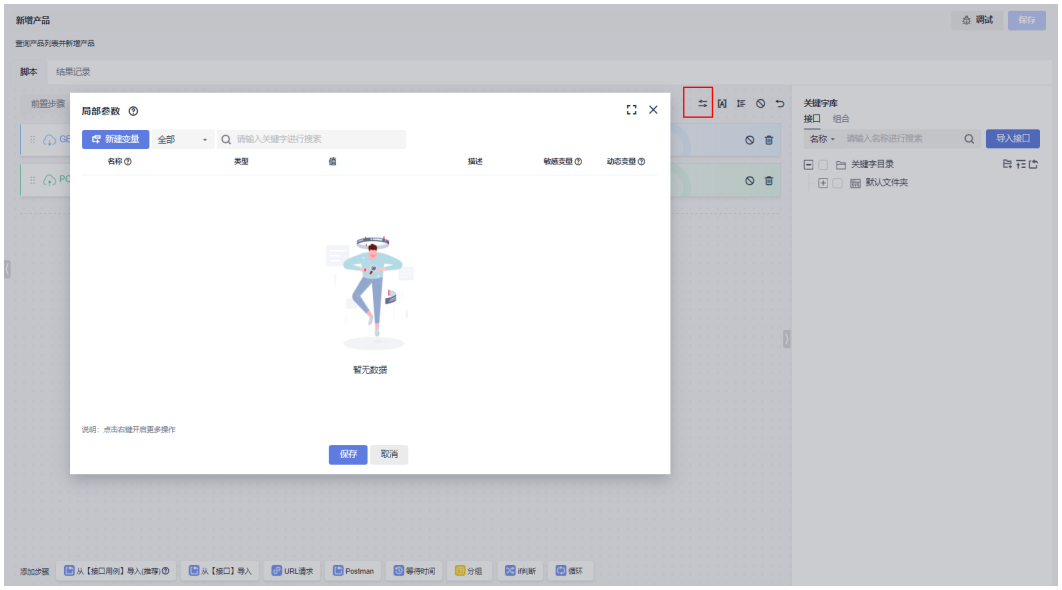
全局环境&变量

请参考**4 环境与环境变量**。

局部参数

局部参数使用范围是当前测试用例，如：测试步骤的参数、检查点、变量等都可以引用局部参数。

局部参数引用形式为“${参数名}”，如：参数名为“hostIp”，则可以使用“${hostIp}”来引用该参数。



局部参数主要配置项如下：

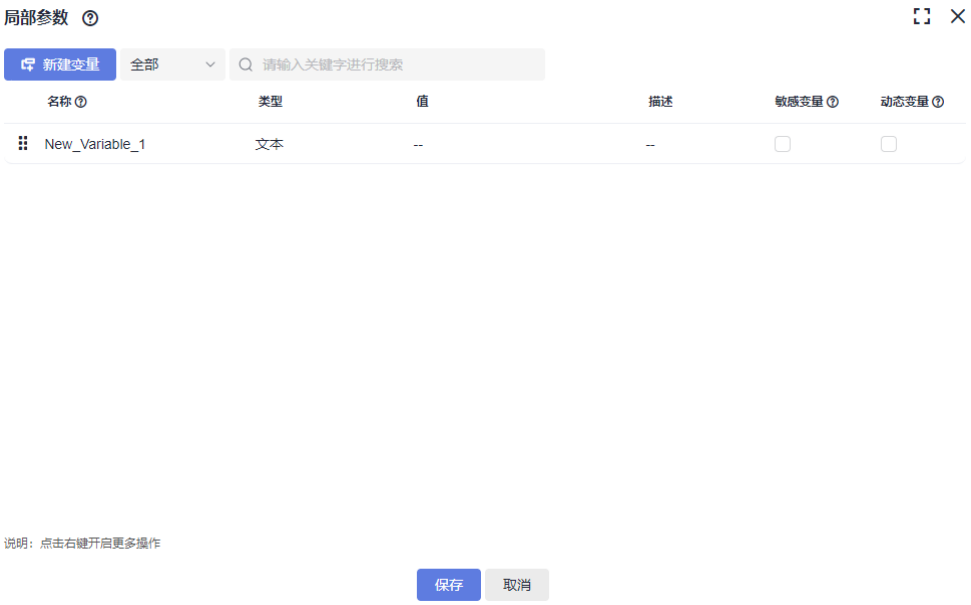
配置项	是否必填	描述
名称	是	支持中英文、数字、点号、中文短横线、下划线的半角字符，并且长度范围是1~300。
类型	是	支持文本、随机字符串、随机整数、时间戳、格式化时间戳、生成UUID、Base64编码、MD5哈希值、密码或认证信息、SHA512编码。
描述	否	对参数的简要描述，上限为3000字符。默认为当前参数类型的描述。 单击“文本框”后，用户可以直接输入文本描述。 单击，输入一个json，单击“Json转换”，可以给没有换行的Json添加换行和缩进，单击“回填”添加成功。
值	否	用户可以为不同类型的参数赋值。
敏感变量	否	勾选为敏感数据后，测试计划对参数值加密存储，在测试结果日志中使用星号覆盖处理。敏感数据类型适用并不限于个人信息、鉴权信息，例如姓名、地址、用户名等。
动态变量	否	动态参数的值可以在用例执行过程中被赋值。动态参数初始值可为空，被赋值之后，此处显示的是最新值。 动态参数赋值方法：在用例测试步骤“响应提取”的“赋值给环境动态参数”中设置后，在测试执行时，响应提取的内容将被赋值给动态参数。

参数类型：

名称	描述
文本	上限为10000字符，支持设置“敏感参数”、“动态参数”，默认为否。
随机字符串	随机生成指定长度字符串，随机字符串的变量值长度校验范围【1-9999】，不支持设置“敏感参数”及“动态参数”。
随机整数	随机生成指定范围整数，区间范围校验为【-999999999~999999999】，不支持设置“敏感参数”及“动态参数”。 例如：设置【-9999,9999】，会获取这个区间内的随机整数。
时间戳	无需输入，生成当前整数时间戳，不支持设置“敏感参数”及“动态参数”。
格式化时间戳	格式为yyyy-MM-dd HH:mm:ss或yyyy-MM-dd，不支持设置“敏感参数”及“动态参数”。 例如： ● yyyy-MM-dd HH:mm:ss：33250825252000，预期值为3023-09-05 20:20:52。 ● yyyy-MM-dd：33250825252000，预期值为3023-09-05。
生成UUID	无需输入，不支持设置“敏感参数”及“动态参数”。
Base64编码	使用Base64方法编码参数，上限为256字符，不支持设置“敏感参数”及“动态参数”。
MD5哈希值	使用指定参数生成MD5哈希值，上限为256字符，不支持设置“敏感参数”及“动态参数”。
密码或认证信息	上限为256字符，不支持设置“敏感参数”及“动态参数”。默认勾选“敏感参数”。
SHA512编码	上限为256字符，不支持设置“敏感参数”及“动态参数”。默认勾选“敏感参数”。
数组	数组的内容是json数组格式，上限为10000字符，不支持设置“敏感参数”及“动态参数”。

- 局部参数管理

1. 进入接口自动化用例编辑的“脚本”页签，进入“局部参数”管理页面。
2. 单击左上方“新建变量”，输入参数名、类型、值，全部参数设置完毕，单击“保存”，完成局部参数设置。



内置参数

内置参数是将HTTP/HTTPS响应的对应部分参数化，在检查点、响应提取功能中的“来源”选项中可以选择内置参数。

内置参数如下表：

内置参数	参数说明	是否支持多级取值	用途	举例
响应体（json）	表示接口返回的响应体。	是	• 检查点的属性字段 • 参数传递的属性字段	• 检查点：判断响应体中的id等于100。 • 设置方法：设置检查点来源为响应体（JSON），属性为id（前提条件响应体JSON串有id字段），设置比较符为等于（字符串、不区分大小写），设置目标值为100。
响应头	表示接口返回的响应头。	是	• 检查点的属性字段 • 参数传递的属性字段	• 检查点：判断响应头中的token等于abcd。 • 设置方法：设置来源为响应头，属性为token（前提条件响应头有token），设置比较符为等于（字符串、不区分大小写），设置目标值为abcd。

内置参数	参数说明	是否支持多级取值	用途	举例
响应码	表示接口返回的响应码。	否	• 检查点的属性或值字 • 变量的属性字段	• 检查点：判断响应码等于200。 • 设置方法：设置来源为响应码，设置比较符为等于（数字），设置目标值为200。

说明

内置参数支持多级取值，例如：

响应体为“{"result":{"appId":12}}”时，则取appId的格式为：来源选择响应体，属性为“result.appId”。如果result是数组格式。则属性为“result[i].appId”，其中，i为非负整数。

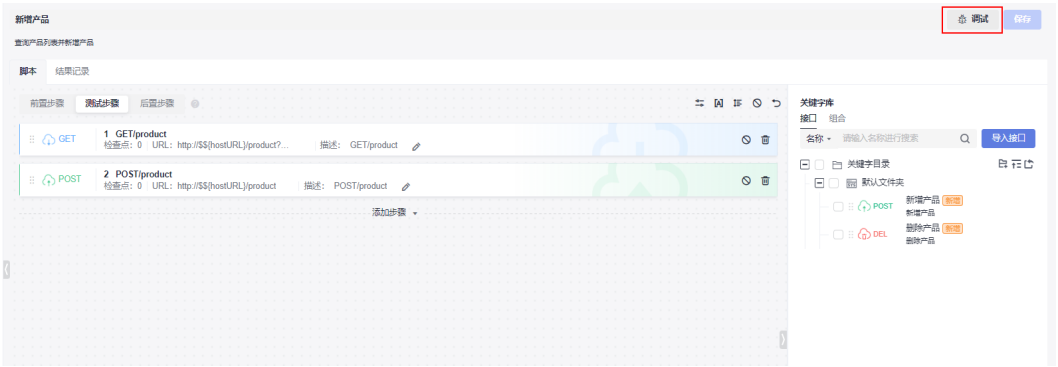
响应提取参数

响应提取参数是从接口的响应体中提取出来的参数，定义及使用方法请参考7.2.4 响应提取。

7.2.8 运行&报告

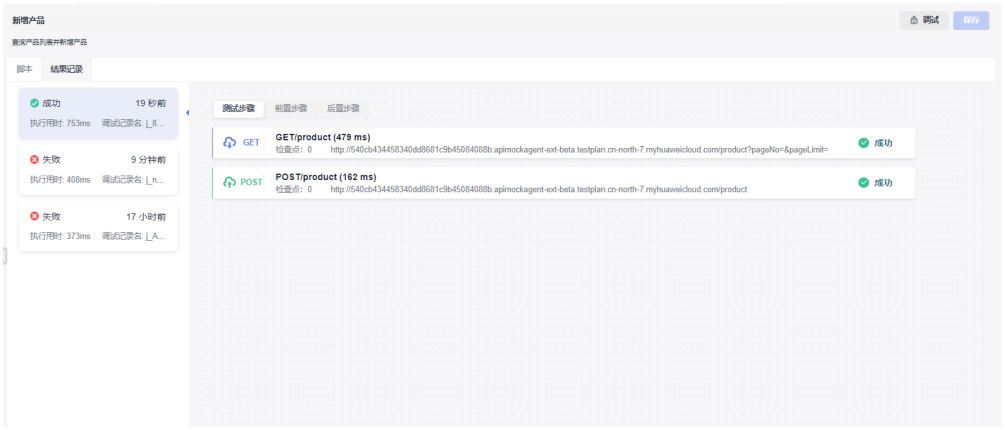
用例调试

在“测试用例”页面，单击“调试”，开始执行测试用例。



查看运行结果

- 调试完成后，可快速查看具体用例执行成功或失败的用时。



- 单击单条结果记录，可查看执行的详细信息。

7.2.9 测试文件夹管理

单击测试用例的任意文件夹，可以查看文件夹下面的全部测试用例。

- 单击用例操作列的▷，可快捷执行测试用例。

- 单击用例操作列的，可快捷查看测试用例执行历史。

- 单击用例操作列的 ，可单击“复制”或“删除”测试用例。

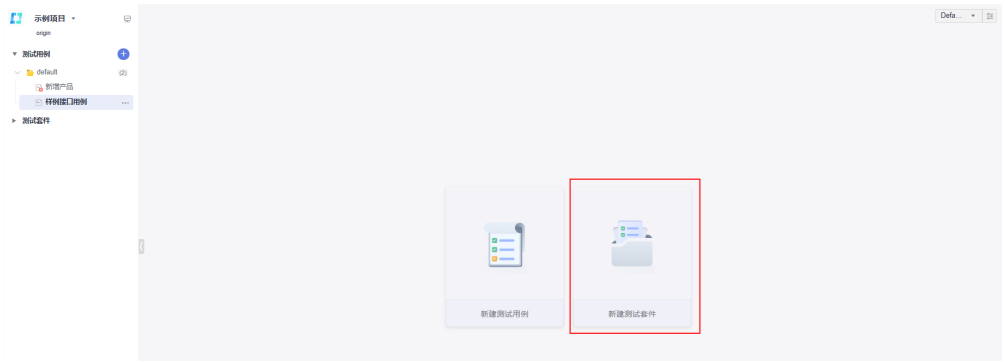
- 全选测试用例，单击“删除”，可批量删除测试用例。

7.3 测试套件管理

7.3.1 创建测试套件

自动化测试套件，实现用例串行/并行执行的策略，测试用例分块加速能力，有效提高资源池利用率，减少任务阻塞情况。并且可提前拦截产品缺陷，更加快速地发现问题。

1. 在“API测试”主页，单击“新建测试套件”。



2. 在“新建测试套件”页面，填写用例名称与描述。

3. 单击“添加测试用例”，弹出“添加测试用例”对话框，选择需要添加到测试套件的测试用例，单击“确定”。

4. 单击“执行策略设置”，弹出“执行策略”对话框，根据需要配置执行策略，单击“确定”。

- 定时类型：执行一次、周期性重复执行，周期性指设置一个执行频率，测试套按照这个频率周期重复执行。
- 任务开始时间：立即执行、指定开始时间。
- 执行时间区间：全天执行、指定执行区间，即指定套件执行的时间段。

- 用例超时时间：设置每个用例的最长执行时间，超过时间，用例则超时失败。 任务继续执行，直到最后一个用例执行完毕。可设置分钟级，小时级，天级。

5. 回到当前测试套件页面，单击右上角“保存“，完成自动化测试套件创建。

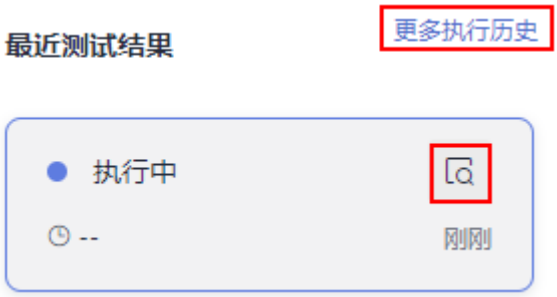
7.3.2 运行&报告

- 运行测试套件

  在待执行测试套件详情页面，单击“执行”，运行测试套件。

- 查看测试报告

  a. 页面右侧可快速查看最近测试结果。



  b. 单击测试结果的  ，可查看测试结果的运行报告。

  c. 单击”更多执行历史”，可查看历史测试结果的运行报告。

8 API 发布

8.1 API 中心发布

介绍如何将已开发好的API分享至API中心。

前提条件

已完成服务商入驻。

操作步骤

1. 单击项目名称，进入目标项目详情页面，单击“API发布 > API中心发布”。

2. 单击“申请发布”，跳转至“创建API资产”页面。
3. 选择导入API、填写API资产信息，参数说明请分别参见**表8-1**、**表8-2**。

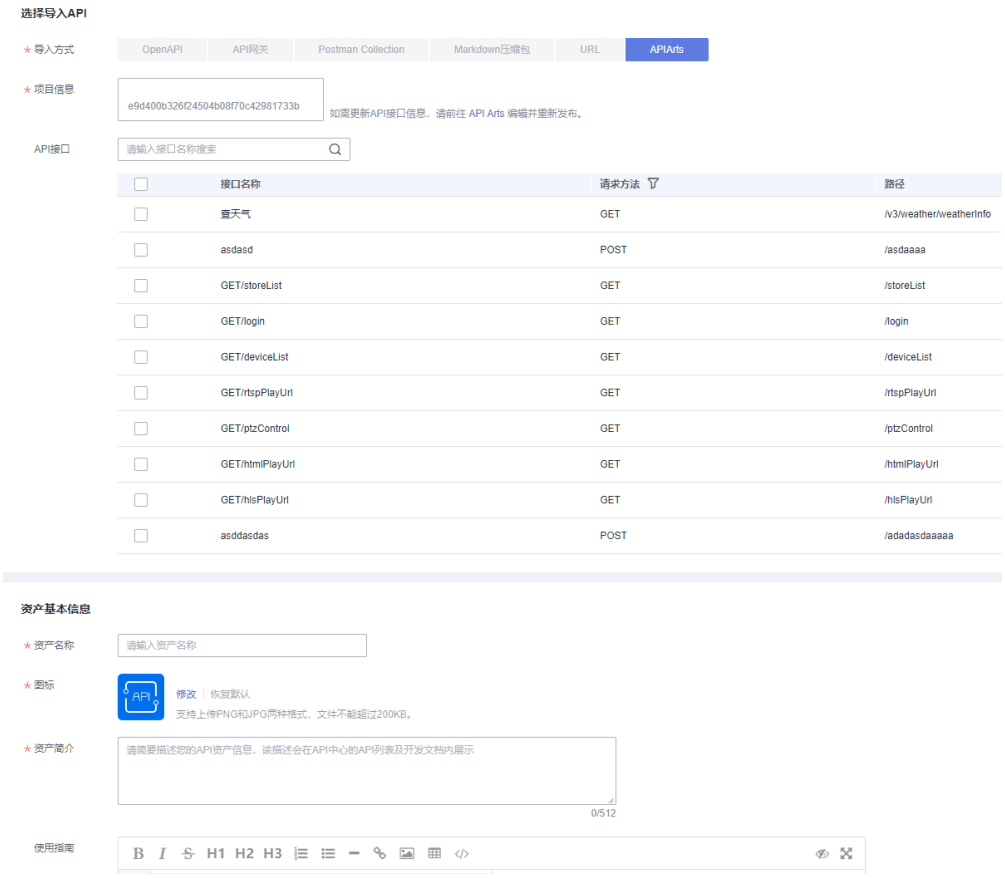


表 8-1 选择导入 API

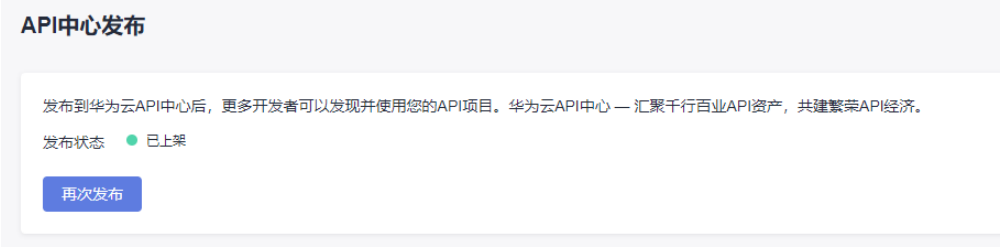
参数名称	说明
导入方式	CodeArts API：支持从CodeArts API工具直接把开发好的API文件一键式导入API中心。 **说明** 从CodeArts API工具导入API资产时，导入方式会默认为“CodeArts API”。
项目信息	CodeArts API项目名称和ID。
API接口	当导入方式为“CodeArts API”时，支持勾选部分接口导入，如果导入全部接口，则忽略此参数。

表 8-2 资产基本信息

参数名称	说明
资产名称	自定义API资产的名称。
图标	为API资产设置一个图标。
资产简介	API资产的简单介绍，方便资产在API中心门户网站按描述被搜索。

参数名称	说明
使用指南	API资产的使用指导，帮助API调用者了解API的使用方法。
资产域名	API资产的域名。
分类	选择资产行业分类，方便API资产在API中心门户网站按行业类别被搜索。
可见范围	• 公开：API资产创建成功后，所有用户都能在API中心门户网站中看到该API资产。 • 私有：API资产创建成功后，仅当前用户能在API中心门户网站中看到该API资产。
API来源	API的来源，方便资产在API中心门户网站按来源被搜索。
标签	为该API资产添加分类属性，方便在大量创建API资产后，快速过滤和查找。

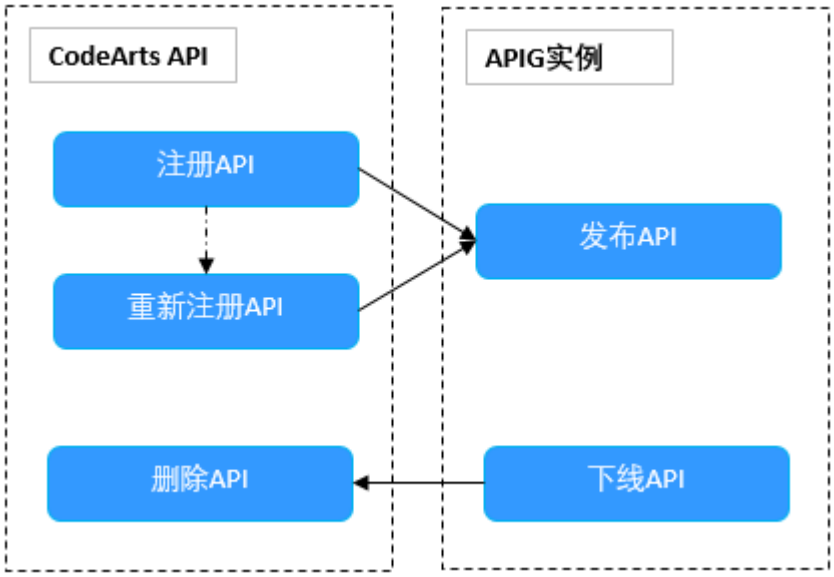
4. 填写完API资产信息后，单击“立即创建”。
5. 等待系统解析API文件，API资产所在行“状态”列如果显示“已上架”，说明解析完成，此时，API资产已成功分享至API中心门户网站。



8.2 API 网关注册

CodeArts API支持将状态为“已测完”或“已发布”的API注册到APIG网关实例，用户需要登录APIG实例，将已注册的API发布，API即可在APIG中正式生效。

API从注册到发布的管理流程如下：



准备工作

在使用API网关注册功能前，用户需要先购买APIG实例。如果没有购买过APIG实例，可以在CodeArts API上通过以下操作步骤完成购买。

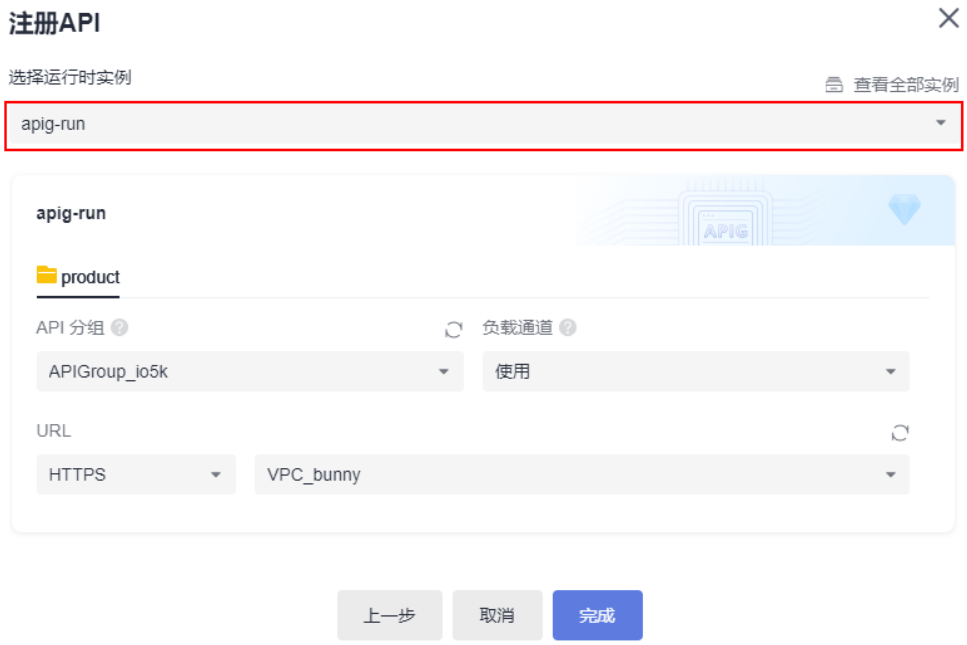
1. 单击“项目设置 > 插件管理”，进入“插件管理”页面。
2. 单击“购买运行时实例”，在弹出的填写购买实例页面，配置实例参数，具体可参照**购买实例**章节，完成购买。

注册 API

1. 单击“API发布 > API网关注册”进入“API网关注册”页面。

2. 单击“注册API”，弹出“注册API”对话框，可以选择项目中所有“已测完”或“已发布”状态的API。

3. 选中需要发布的API，单击“下一步”。
4. 在“选择运行时实例”的下拉框中选择需要注册的APIG实例，并配置实例参数。参数配置请参考APIG实例参数表。



APIG实例参数

参数	配置说明
API分组	在APIG实例上已创建的API集合，创建操作可参照**创建API分组**章节。
负载通道	是否使用负载通道。 ● 选择“使用”时，需要在URL中设置使用的负载通道。如未创建负载通道时，可单击“创建负载通道”加号按钮，直接跳转“新建负载通道”页面完成创建，具体创建操作可参照**负载通道**章节。 ● 选择“不使用”时，需要在URL中设置后端服务地址。
URL	URL地址由请求协议、负载通道/后端服务地址组成。 ● 请求协议：HTTP或HTTPS，传输重要或敏感数据时推荐使用HTTPS。 ● 负载通道（可选） 仅在使用负载通道时，需要设置。选择已创建的负载通道名称。 ● 后端服务地址（可选） 仅在不使用负载通道时，需要设置。

5. 单击“完成”，完成API在APIG实例中的注册。

说明

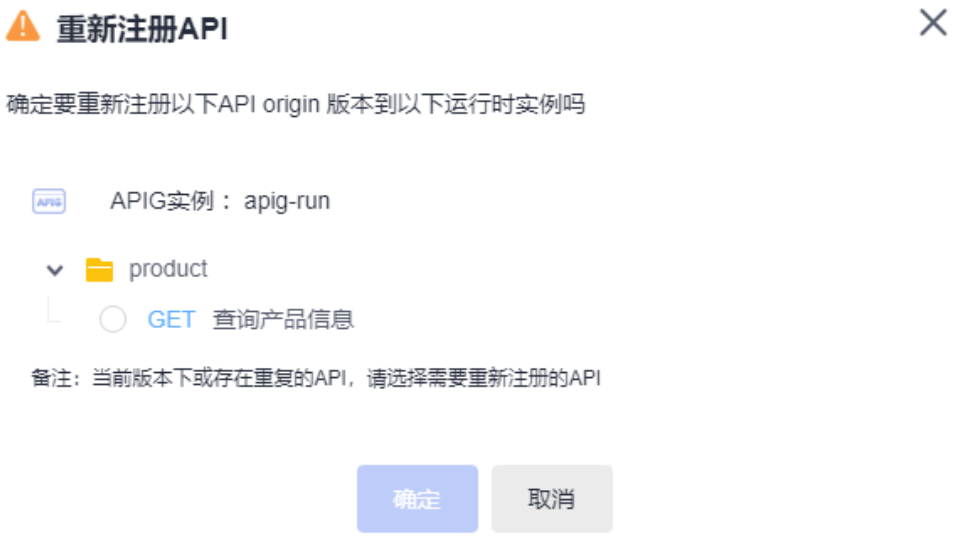
- 一个API可发布到同一个APIG的不同API分组。
- 对于项目中同名API的处理：如果项目中存在多个同名的API（URL、请求方式、名称相同），如果已经注册同名API到APIG实例的第一个分组中，则二次注册同名API会失败。APIG实例的同一个分组上不允许同名API重复注册。
- 注册API后需要到APIG实例中完成**发布API**操作，才可完成API的上线。

API 重新注册

CodeArts API提供API重新注册的功能，协助用户将已修改或不同版本的API重新注册到APIG实例。

1. 在“API网关注册”页面，光标移动到API版本，可查看当前API的注册信息。

2. 单击操作列的，弹出“重新注册API”对话框。



3. 选择需要重新注册的API，单击“确定”，完成API的重新注册。

说明

- 重新注册的API需要到APIG实例中完成**发布API**操作，才可完成API的上线。
- 对于项目中同名API的处理：如果项目中存在多个同名的API（URL、请求方式、名称相同），在重新注册操作时，需要用户从多个同名API中选择一个重新注册。APIG实例的同一个分组上不允许同名API重复注册。

API 删除

1. 在“API网关注册”页面，选择需要删除的API，单击“删除”。

2. 弹出“删除警告”对话框，单击“确定”，完成API的删除。
   删除失败会弹出“删除失败”对话框，提示失败原因。

说明

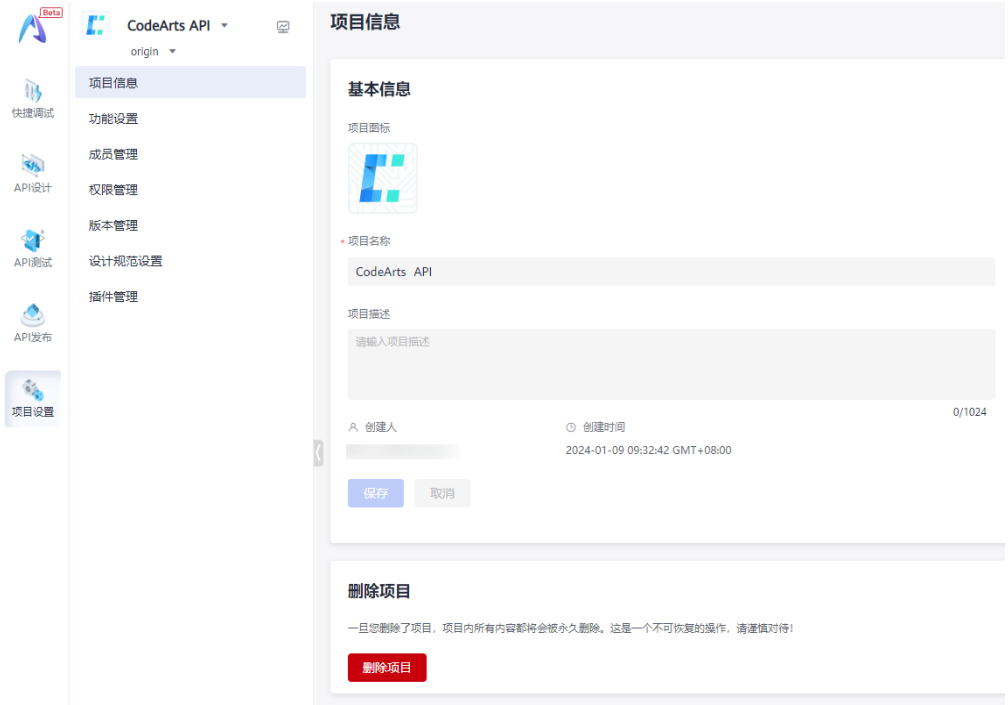
删除在APIG实例中已发布的API，需要先在APIG实例中进行**下线API**操作，才可完成API的删除。

9 项目设置

项目管理为开发团队提供简单高效的团队协作服务，用于创建、管理和使用软件开发项目。包含项目管理、成员管理、权限管理、设计规范设置、插件管理等功能。

9.1 项目信息

单击项目名称，进入目标项目，单击“项目设置 > 基本信息”，可以修改项目基本信息，还可以对项目进行删除操作。



修改项目基本信息

1. 在“基本信息”模块，可以根据实际需要修改项目名称、项目描述等。
2. 修改完成后，单击“保存”。

说明

创建时间不支持修改。

删除项目

在“删除项目”模块，单击“删除项目”，输入提示信息后单击“确定”。

说明

删除项目会删除项目下的全部资源，且数据无法恢复，请慎重删除。

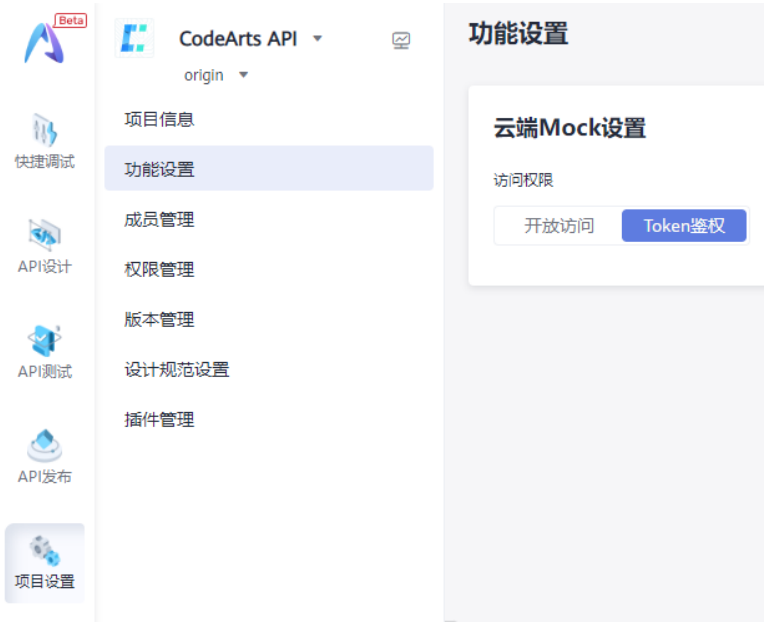
9.2 功能设置

CodeArts API默认开启云端Mock，云端Mock功能适用于团队的共同协作场景，提供一个固定地址，其他成员可以通过这个地址访问云端Mock Server。

云端Mock的优势：

- Mock数据统一管理
- 配置数据团队共享

单击项目名称，进入目标项目，单击“项目设置 > 功能设置”，选择“开放访问”或“Token鉴权”，即可设置云端Mock访问权限。



9.3 成员管理

介绍如何为项目添加成员、设置项目成员角色以及将成员移出项目（删除项目成员）。

前提条件

- 对项目具有管理权限。
- 项目管理员才可以添加成员或将成员移出项目。
- 项目创建者可以删除项目经理和测试经理。
- 项目经理可以删除测试经理。

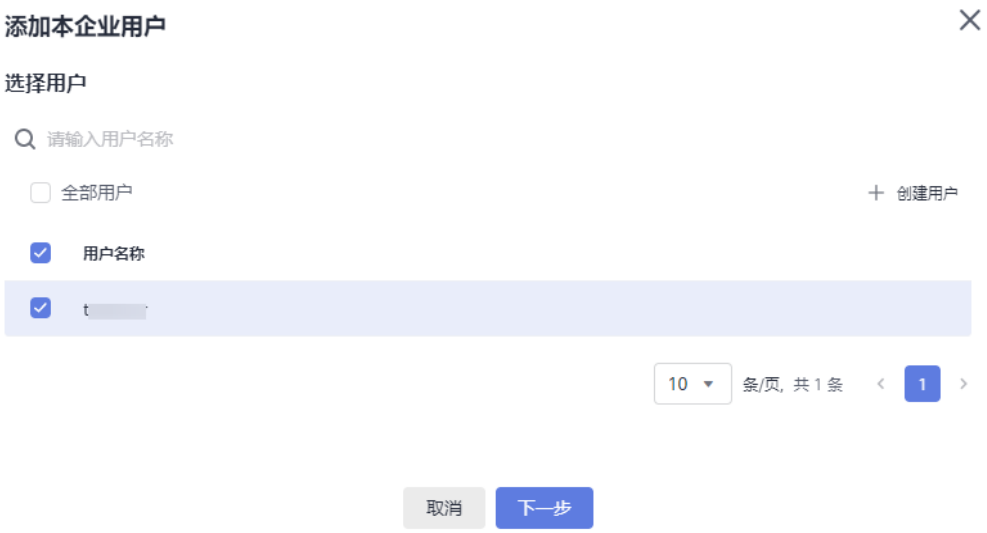
操作入口

单击项目名称，进入目标项目，单击“项目设置 > 成员管理”，进入“成员管理”页面。

添加本企业用户

本企业用户为企业用户在统一身份认证服务中已创建的IAM用户，即成员下显示的用户。

1. 单击“添加成员”旁的下拉框，选择“添加本企业用户”。
2. 弹出“添加本企业用户”对话框，勾选需要添加的成员，然后单击“下一步”。



3. 选择项目角色，然后单击“完成”即可成功添加。

从其他项目导入用户

通过导入其他项目下的用户，可以快速完成项目成员的添加。

1. 单击“添加成员”旁的下拉框，选择“导入其它项目用户”。
2. 弹出“导入其他项目用户”对话框，在“项目源”下拉列表框中选择已有项目（项目支持搜索）。

3. 单击“导入”，可以将其他项目下的成员导入到当前项目中。

邀请其他企业用户

可以通过邀请其他企业用户，对当前项目进行查看或编辑。

1. 单击“添加成员”旁的下拉框，选择“邀请其他企业用户”。
2. 弹出“邀请其他企业用户”对话框，在“企业用户”输入框中输入企业用户ID，在“用户名称”输入框中输入需要邀请的用户名称后，单击右侧“添加用户”，可一次性添加同一企业不同用户。

3. 单击“确定”导入其他企业成员，默认为“浏览者”身份权限。可通过**设置项目成员角色**修改成员角色与权限。

批量添加项目成员

如果添加成员数较多，可以批量添加项目成员。

1. 管理员通过“统一身份认证服务”创建用户。
2. 在“成员管理”页面，单击“添加成员”旁的下拉框，选择“添加本企业用户”。
3. 选择全部要添加的成员，单击“完成”。即可完成项目成员的批量添加。

设置项目成员角色

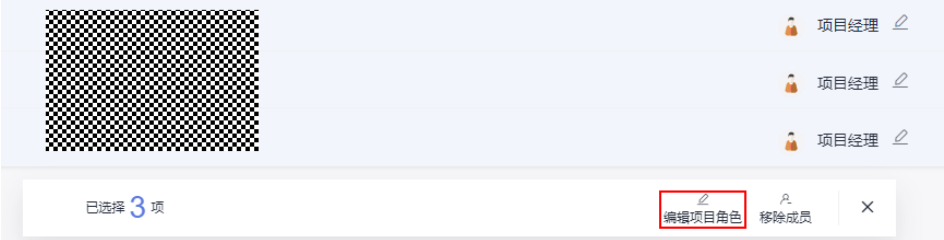
项目创建者、项目管理员（项目经理和测试经理）才可以给成员设置项目角色，项目角色不同，各服务的访问权限也不同。默认的项目角色类型如下：

- 项目创建者
- 项目经理
- 开发人员
- 测试经理
- 测试人员
- 参与者
- 浏览者
- 运维经理

项目成员添加完成后，可以给项目成员设置不同角色。

1. 在“成员管理”页面，将鼠标移动到项目成员角色类型旁，单击编辑按钮。

2. 弹出“编辑项目角色”对话框，根据实际情况为成员设置项目角色，单击“确定”完成角色设置。
3. （可选）批量修改项目成员角色。
   a. 在“成员管理”页面，勾选目标成员，然后单击“编辑项目角色”。



   b. 在弹出的“批量修改成员角色”对话框中修改角色类型。
   c. 选择需要更改的项目角色。
   d. 单击“确定”，即可完成项目成员角色批量修改

移出成员

可以移出单个成员，也可以批量移出多个成员。

- 移出单个项目成员

  在“成员管理”页面，单击目标成员所在行的  ，根据提示完成移出。
  - 成员被移出后，即该用户被移出项目。
  - 成员被移出后，其名下的工作项等资源不会被删除，用户操作记录仍旧会被保留。
- 批量移出项目成员
  a. 在“成员管理”页面的项目成员列表中勾选目标成员。
  b. 单击“移除成员”。

  c. 根据提示单击“确定”，即可完成项目成员批量移出。

9.4 权限管理

为项目各个角色设置不同操作权限，包括项目基本信息的编辑和归档、角色及权限的设置和成员设置，还可以根据需要自定义角色类型并授权。

1. 单击项目名称，进入目标项目，单击“项目设置 > 权限管理”，进入“权限管理”页面。
2. 选择需要修改的角色名称和选项，在右侧勾选设置不同操作权限。

3. 单击“新增角色”，可以自定义角色类型并设置相关权限。

9.5 版本管理

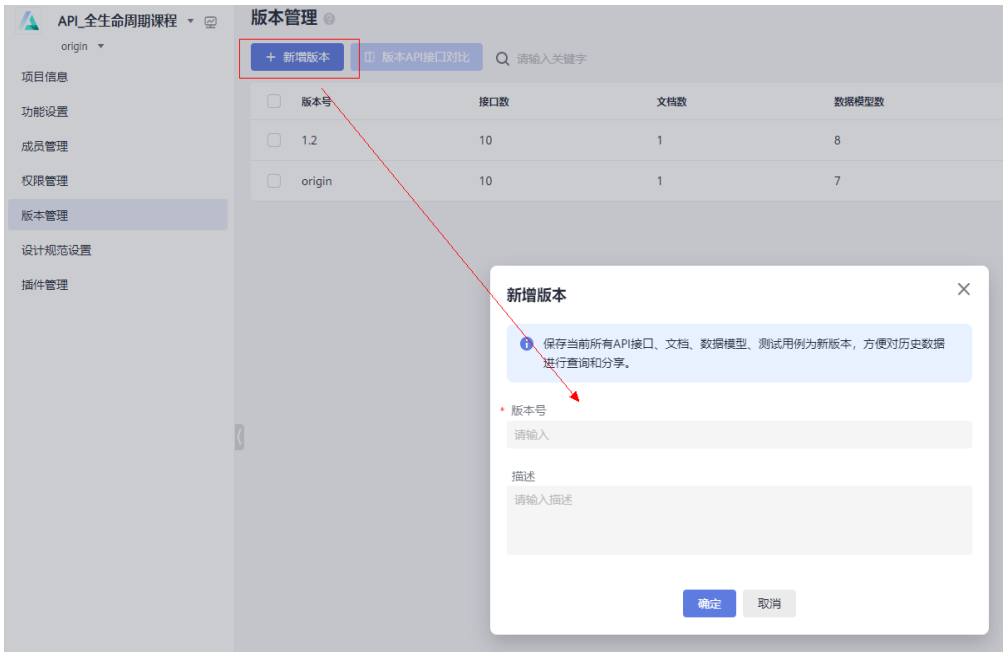
CodeArts API提供API设计版本管理功能，可在不同版本之间进行快速切换，也可对不同版本API接口进行直接对比。

说明

目前只支持API设计模块中API接口版本新增，暂不支持API测试模块相关测试用例与测试套件版本同步。

创建版本

1. 单击项目名称，进入目标项目，单击“项目设置 > 版本管理”，进入“版本管理”页面。
2. 单击“新增版本”，填写版本号和描述。



3. 单击“确定”。
4. 单击项目名称下“origin”旁的下拉框，可快速切换版本（支持根据版本名快速搜索），并对版本进行编辑。

克隆成项目

CodeArts API统一项目不同版本支持单独克隆为新项目。

说明

目前只支持API设计模块中的API接口设计克隆为新项目中API设计模块的接口设计，API测试模块暂不支持克隆当前API测试模块相关测试用例与测试套件。

1. 在“版本管理”页面。
2. 单击目标版本所在行的，弹出“克隆成项目”对话框。

3. 输入需要克隆成项目版本的项目名称与项目描述，单击“确定”。

版本对比

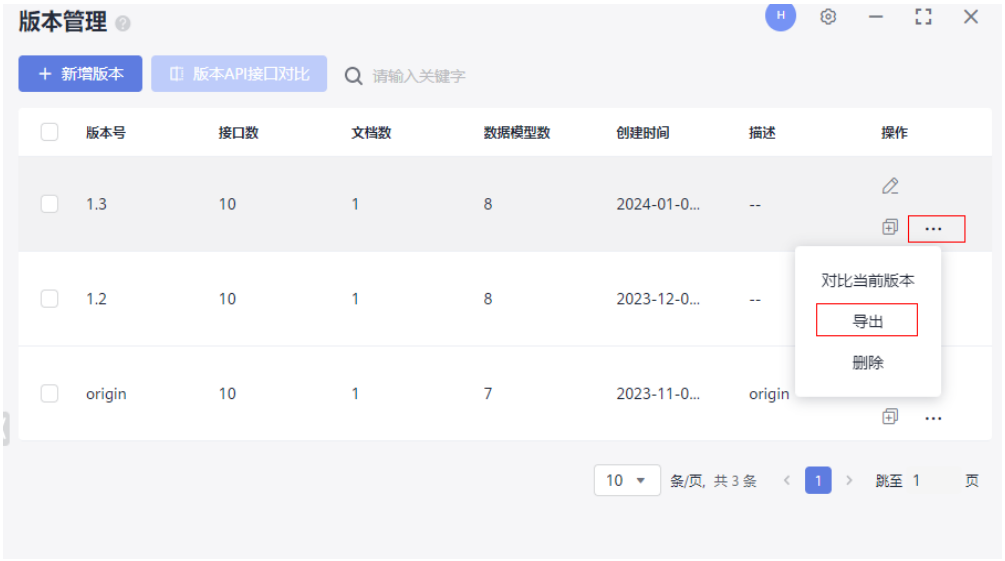
CodeArts API提供同项目内不同版本的API对比功能。

在“版本管理”页面，选择需要对比的两个版本，单击“版本API接口对比”，可对比不同版本的具体内容和变更细节。

接口导出

CodeArts API提供不同版本接口的导出功能。

1. 在“版本管理”页面，单击目标版本所在行的 ，选择“导出”。

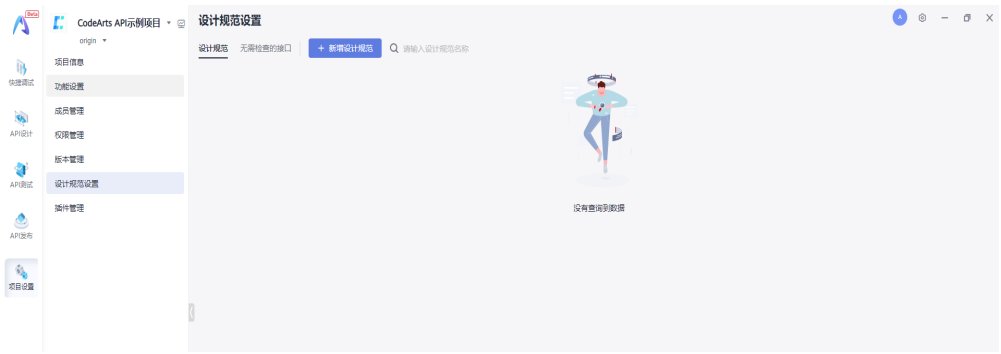


2. 选择文件在本地存储位置、设置文件名，保存到本地，解压后可查看导出的接口文件。

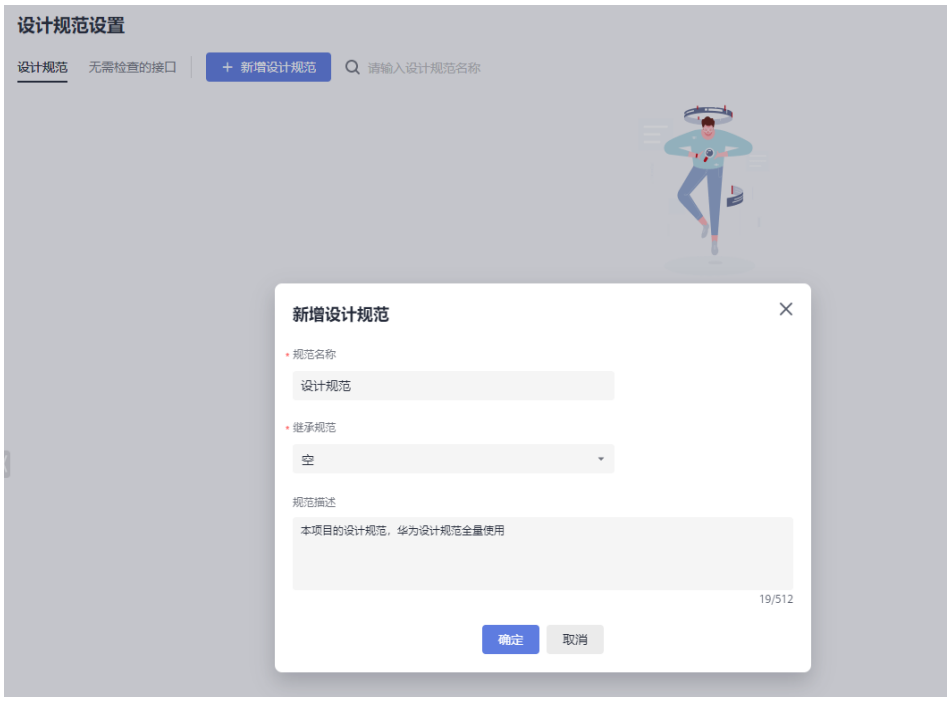
9.6 设计规范设置

CodeArts API提供华为云设计API规范性检查功能。通过规范性检查功能，可以形成统一的设计规范，避免API设计不一致，提高API设计质量。

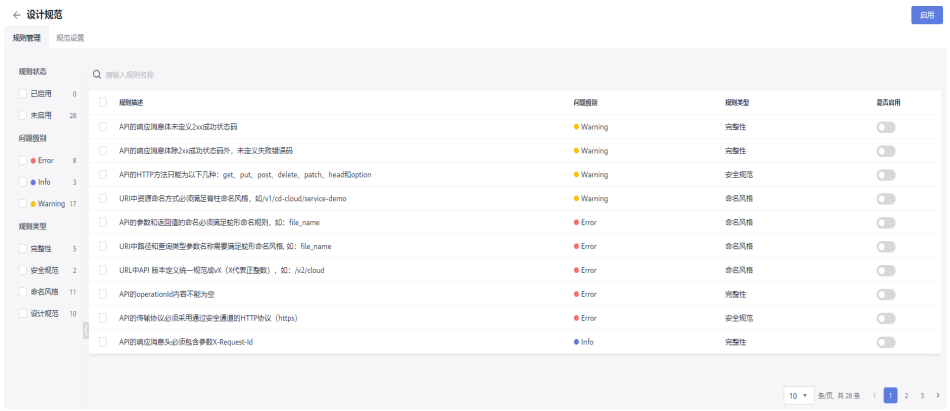
单击项目名称，进入目标项目，单击“项目设置 > 设计规范设置”，进入“设计规范设置”页面。



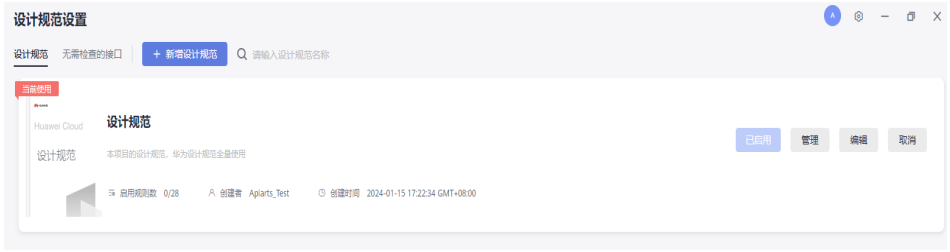
- 新增设计规范
  a. 在“设计规范”页签，单击“新增设计规范”，填写规范名称，选择是否继承已创建的规范、填写规范描述。



b. 单击“确定”，进入“设计规范”页面，打开规则后的“是否启用”按钮，可启用当前规则。



c. 完成创建API设计规范后，单击“启用”，可正式使用设计规范。



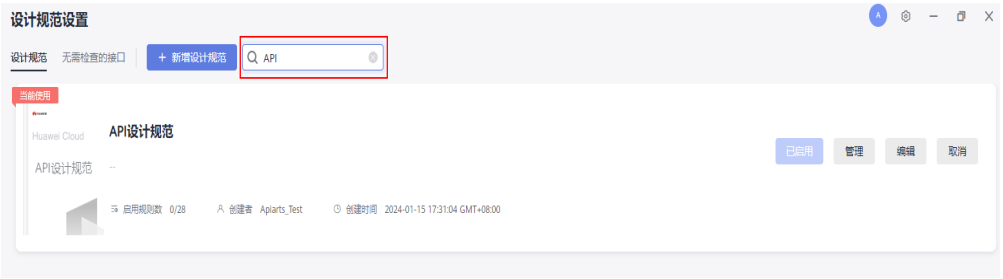
- 无需检查接口

  支持添加无需检查接口。

  a. 在“无需检查的接口”页签，单击“添加API接口”，弹出框添加无需检查接口。

b. 选择无需检查接口，单击“确定”，完成添加。

- 快捷搜索
  可输入关键字快捷查找已存在的设计规范。



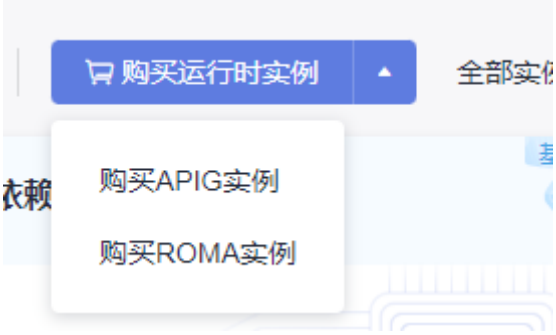
- 设计规范定义更新
  单击“管理”，可以修改设计规范中启用的设计规则。
  单击“编辑”，可以修改设计规范的名称和描述。
- 设计规范删除
  单击“删除”，可以删除设计规范。

9.7 插件管理

在插件管理页面中可以查看APIG和ROMA实例，还可以根据需要购买实例。

单击项目名称，进入目标项目，单击“项目设置 > 插件管理”，进入“插件管理”页面。

- 单击实例名称，可进入实例详情页面，查看实例详情。
- 单击“购买运行时实例”旁的下拉框，选择“购买APIG实例”或者“购买ROMA实例”，根据界面提示购买相应实例，具体操作可参考**购买APIG实例**和**购买ROMA实例**。



10 客户端与插件

10.1 客户端

Windows 平台提供了桌面应用程序可访问 **官网首页** 进行下载，下载完成后解压文件，运行解压文件即可安装 CodeArts API 到您的电脑。

客户端特性

- 登录使用
  单击“登录/注册”，填写账户密码进行登录，即可体验客户端功能。

- 未登录使用

  打开客户端，在登录页面单击“稍后再说”即可开始使用。未登录的状态下只支持创建一个环境及使用部分快捷调试功能。

- 创建环境

  创建环境及环境变量请参照**环境与环境变量**进行设置与使用。

- 快捷调试

  进入项目后，单击搜索框旁 ，可根据需要选择“新建快捷调试”进行**新建快捷调试**操作。

说明

- 导入与导出功能需要登录后才可以继续使用。
- 前置脚本、后置脚本功能需要登录后才可继续使用。

- 切换登录状态

  单击以下功能菜单可弹出登录页面，进行登录后可体验产品的全部功能。

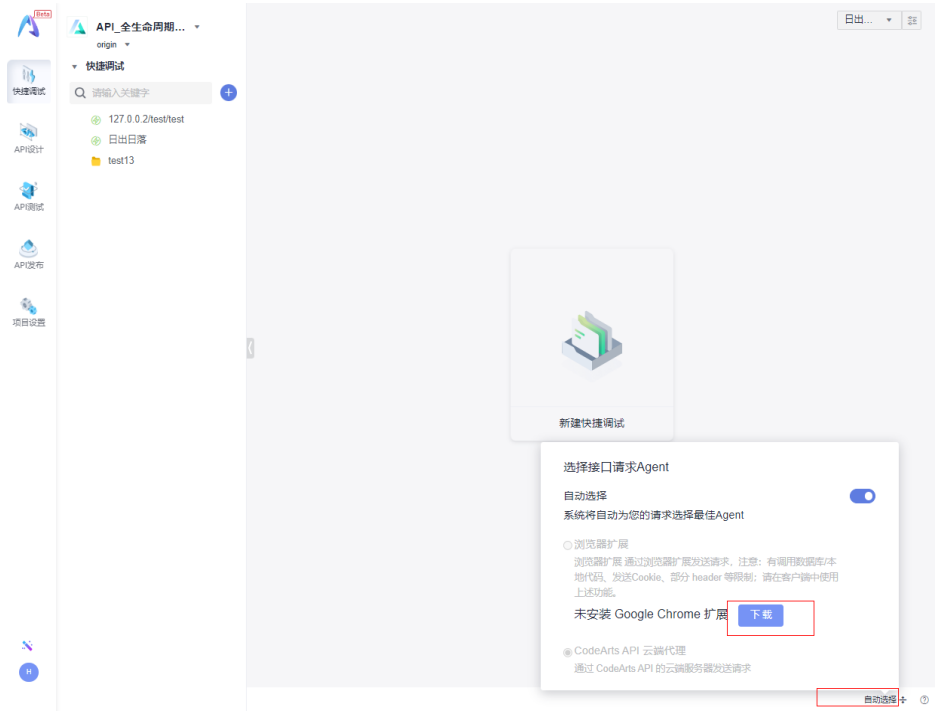
10.2 浏览器扩展

CodeArts API Web版本需要安装浏览器扩展，通过浏览器扩展发送请求。

说明

有调用数据库/本地代码、发送Cookie、部分header等限制；请在客户端中使用上述功能。

步骤1 单击项目名称，进入目标项目后，单击“自动选择 >下载”。



步骤2 下载后解压文件到文件夹中。

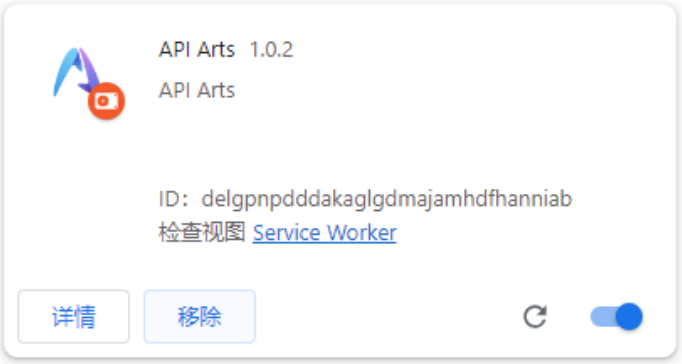
步骤3 在Chrome浏览器设置页面单击左下角“扩展程序”。

步骤4 进入扩展程序页面后，打开右上角“开发者模式”。

步骤5 单击左侧“加载已解压的扩展程序”。

步骤6 加载**步骤2**已解压的文件夹。

步骤7 单击加载后，即可在Chrome浏览器扩展程序中管理已加载的浏览器扩展。

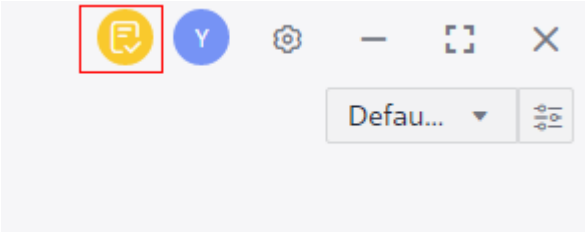


----结束

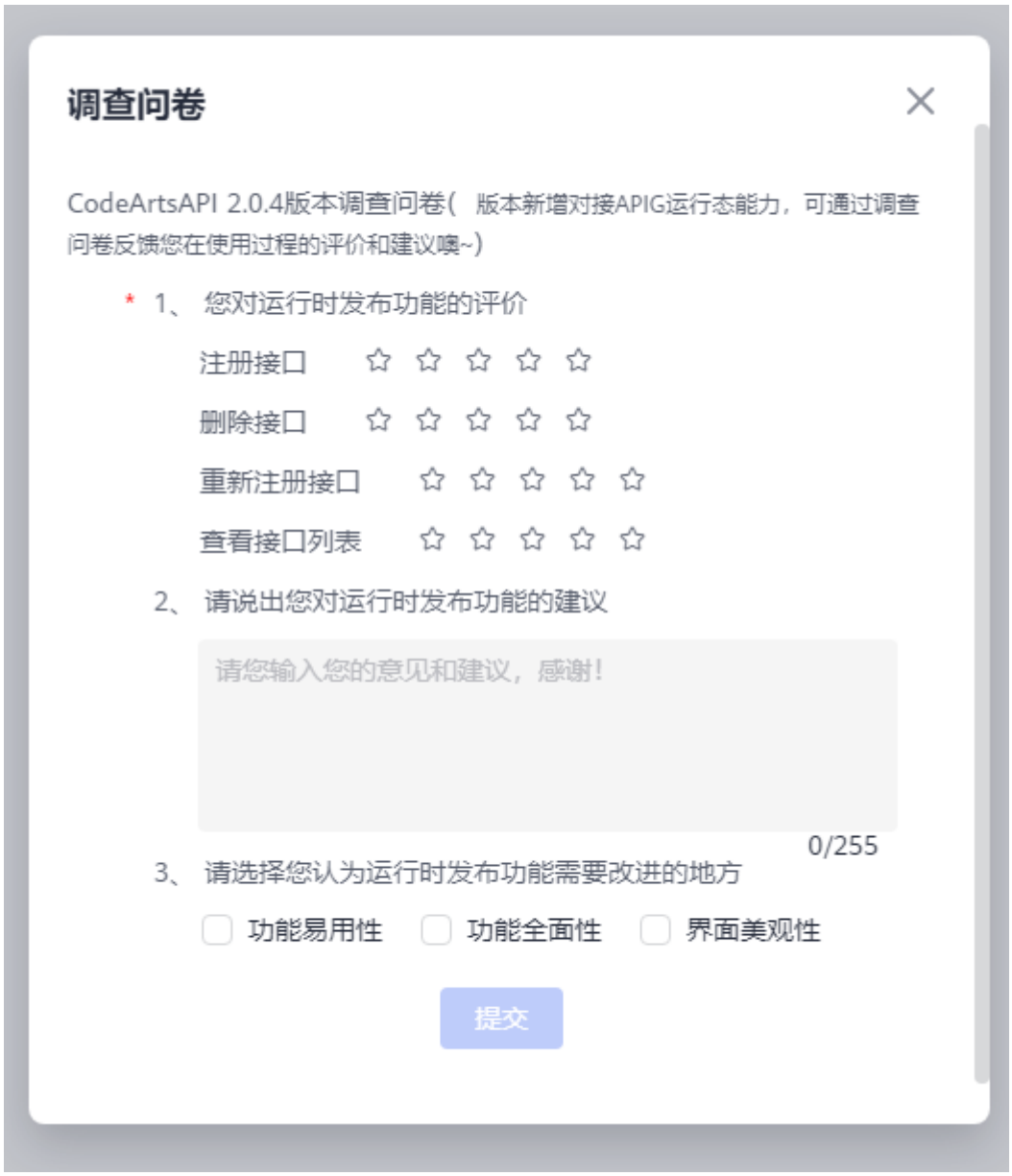
11 调查与问卷

感谢参与我们的调查与问卷，用户的关注是我们前进的动力，可以通过以下步骤参与我们的调查与问卷。

步骤1 当有新特性的调查问卷时，用户可以点击右上角问卷入口参与填写调查问卷。



步骤2 根据用户的使用体验完成问卷内容，单击“提交”即可完成问卷。



----结束

