# token

APS-0NpyZnKw3yfwvM1EcVuhkWG9CLafSISY

# 帮助文档

https://docs.apifox.com/

# 基本概念

## 1、接口设计

定义接口规范（如接口路径、参数、返回值、数据结构等），也就是编写接口文档。

## 2、接口运行

输入特定的接口参数，然后运行，主要用来调试接口。接口运行完之后点击保存为用例按钮，即可生成接口用例，后续可直接运行接口用例，无需再输入参数，非常方便。

## 3、接口用例

通常一个接口会有多种情况用例，比如参数正确用例、参数错误用例、数据为空用例、不同数据状态用例等等。运行接口用例时会自动校验数据正确性，用接口用例来调试接口非常高效。

## 4、数据模型

可复用的数据结构，定义接口返回数据结构及请求参数数据结构（仅 JSON 和 XML 模式）时可直接引用。

## 5、集合测试

多接口集成测试，主要有功能测试、性能测试、对比测试。

## 6、快捷请求

类似 Postman 的接口调试方式，主要用途为临时调试一些无需文档化的接口，无需提前定义接口即可快速调试。

## 7、代码生成

根据接口及数据模型定义，系统自动生成接口请求代码、前端业务代码及后端业务代码。

## 8、Apifox 是区分接口设计和接口运行两个概念的。

接口设计：即新建接口界面或接口详情里的编辑界面，用途是定义接口文档规范，而不是运行接口，所以该界面是只能定义接口基本信息、参数名及参数说明等，而不能设置参数值。参数值、前置脚本/后置脚本 等信息请在接口运行界面或接口用例界面填写。

接口运行：即接口详情里的运行界面，用途是临时调试接口，运行完后，需要点击保存为用例，才能将填写的参数值、前置脚本/后置脚本等信息保存下来；否则关闭 tab 后，这些信息将会丢失。

# 接口管理

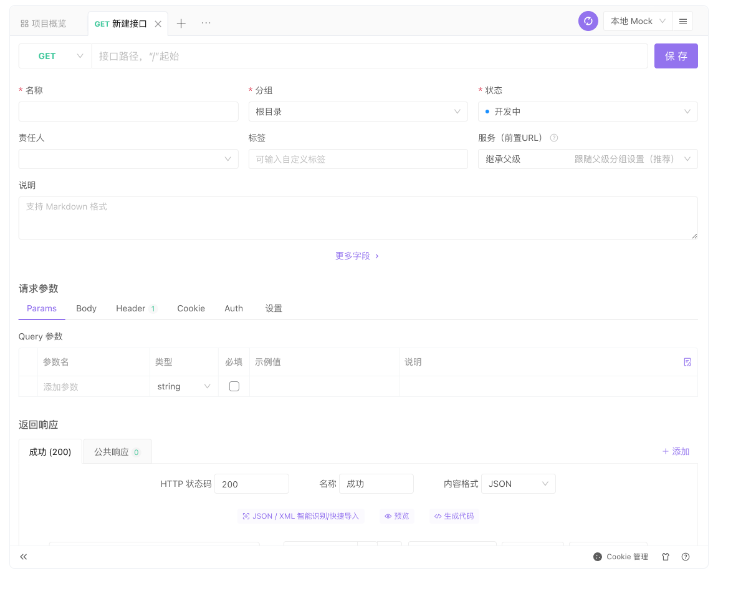
## 接口设计

### 快速上手

1、点击左侧搜索框旁边的 + 号按钮即可打开新建窗口。



2、在打开的窗口中，直接定义接口相关信息。



### 接口路径

以斜杠/起始的接口 path 部分，如/pets、/pets/{id}。

说明：

1、**接口路径** 建议不要包含 HTTP 协议及域名，这部分建议在 [环境管理](https://apifox.com/help/app/api-manage/environments/) 的前置 URL里设置，接口调试时的 URL 会自动加上当前环境的前置 URL。

2、特殊情况需在接口路径要带上HTTP 协议及域名的，系统也能支持，但不建议这么做。接口调试时，系统如检测到接口路径是以http://或https://起始的，会自动忽略当前环境里前置 URL。

3、Apifox 中的 Path 参数是以大括号包裹起来表示，而非冒号起始表示。**正确示例**：/pets/{id}，**错误示例**/pets/:id。

4、**接口路径** 不可包含Query 参数（即 URL 中 ?后的参数），Query 参数在下方请求参数部分填写。

### 请求参数

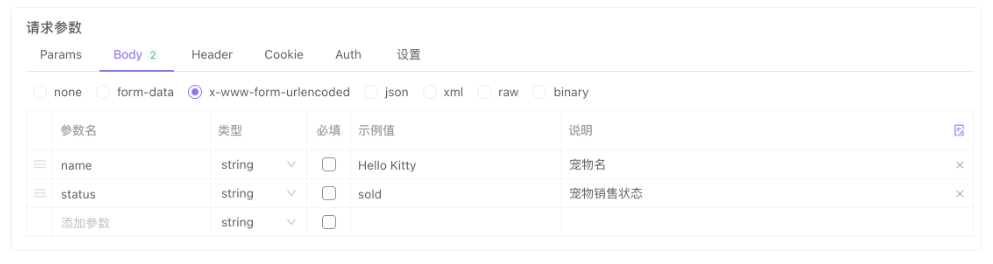
#### Params 参数

包含 Query 参数和 Path 参数两部分。

1、Query 参数：即 URL 中 ?后的参数。

2、Path 参数：自动提取接口路径中大括号包裹起来的参数，如/pets/{id}中的的{id}即表示名为id的 Path 参数。

#### Body 参数



说明：

1、Body 参数类型为json或xml时，需要设置数据结构，并且数据结构可以引用数据模型。

2、接口发送请求的时候会根据Body 参数类型自动在请求Header加上对应的Content-Type，无需手动设置。

### 参数中使用环境变量（或者全局变量/临时变量）

所有参数都可以使用变量，使用方式为双大括号包裹变量名，如{{my\_variable}}，表示引用名为my\_variable的变量。参数值使用变量时可以包含变量以外的字符串，如：参数值设置为prefix-{{my\_variable}}-surfix，假设运行时变量my\_variable的值为123，则实际请求时参数的值为prefix-123-surfix。

### 返回响应



返回响应定义主要包含以下几部分

1、接口返回的 HTTP 状态码

2、返回内容的数据格式：JSON、XML、HTML、Raw、Binary

3、数据结构：仅JSON、XML可配置数据结构

说明：

1、当一个接口在不同情况下会返回不同数据结构时，可设置多个返回响应。点击返回响应模块右上方的+ 新建即可添加。

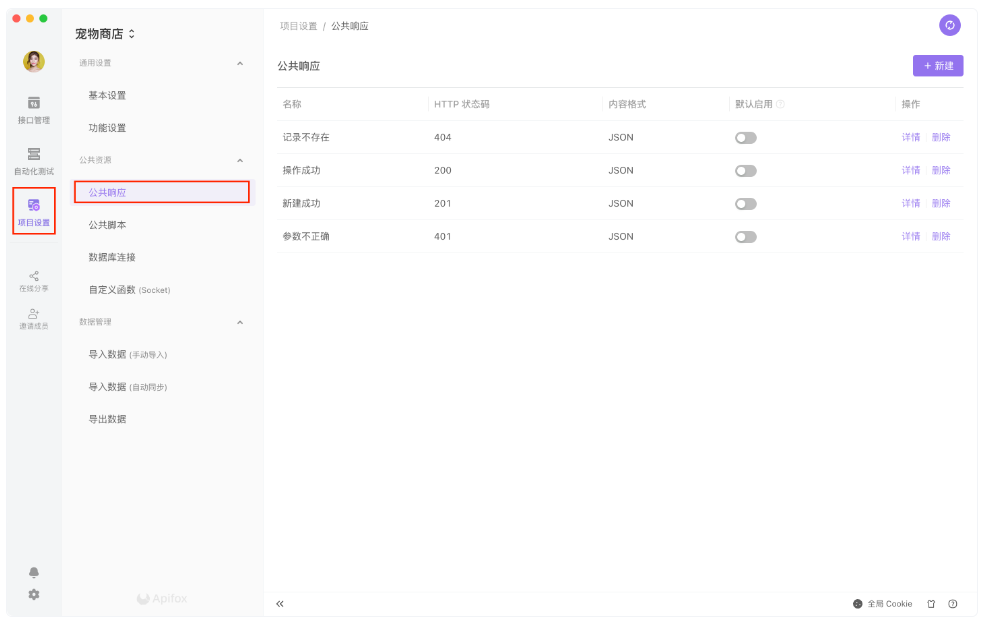
2、定义好数据结构后，接口调试时，系统会自动校验返回的数据是否符合定义的数据结构，非常方便。

3、定义好数据结构后，使用 mock 功能时，系统会自动根据定义的数据结构 mock 出非常人性化的数据，非常方便。

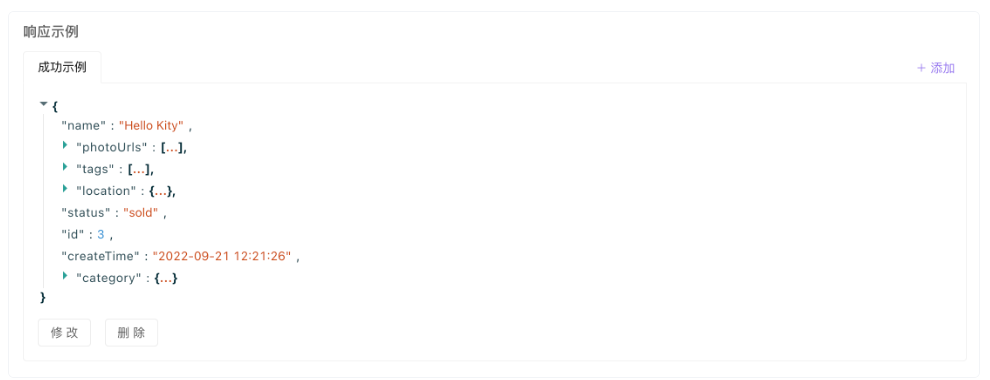
### 公众响应

公共响应主要用来实现返回响应的复用。通常不同接口在某些情况下会返回相同的数据结构，如资源不存在(404)、没有访问权限(401)等，这些建议设置为公共响应，避免重复编写，方便统一管理。

设置方法：打开项目设置->公共响应，在这里管理公共响应。



### 响应示例



设置返回响应的示例数据，方便查阅接口文档的人快速了解数据结构。返回 Response 的示例数据也可以设置多次，点击响应示例模块右上方的+ 新建即可添加。建议至少设置两个示例：成功示例、失败示例。

## 接口调试/接口用例

## 调试模式

## 前置操作/后置操作

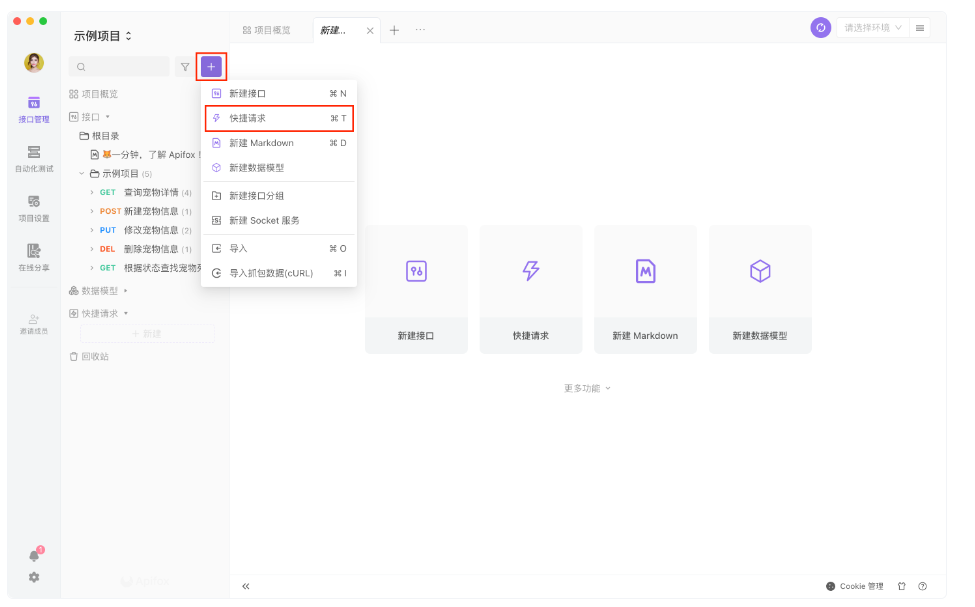
## 数据结构/数据模型

## 快捷请求

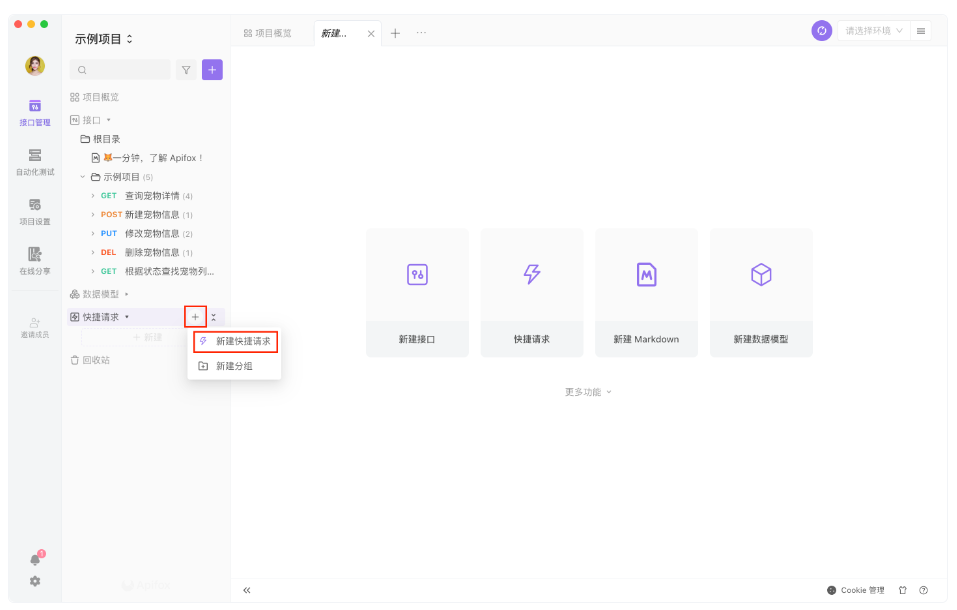
### 使用场景

临时需要调试某个接口，而不需要保存为文档或者用例。

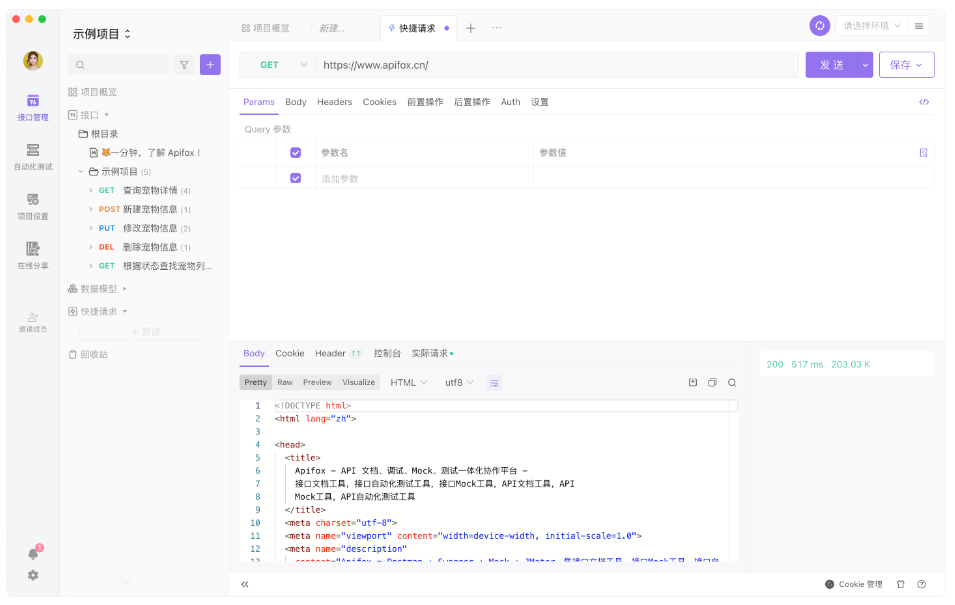
1、鼠标移到左侧搜索框旁边的 + 号按钮，在下拉列表里选择快捷请求。



2、鼠标移动到左侧目录中，在快捷请求右边点击 + 号按钮，也可新建快捷请求

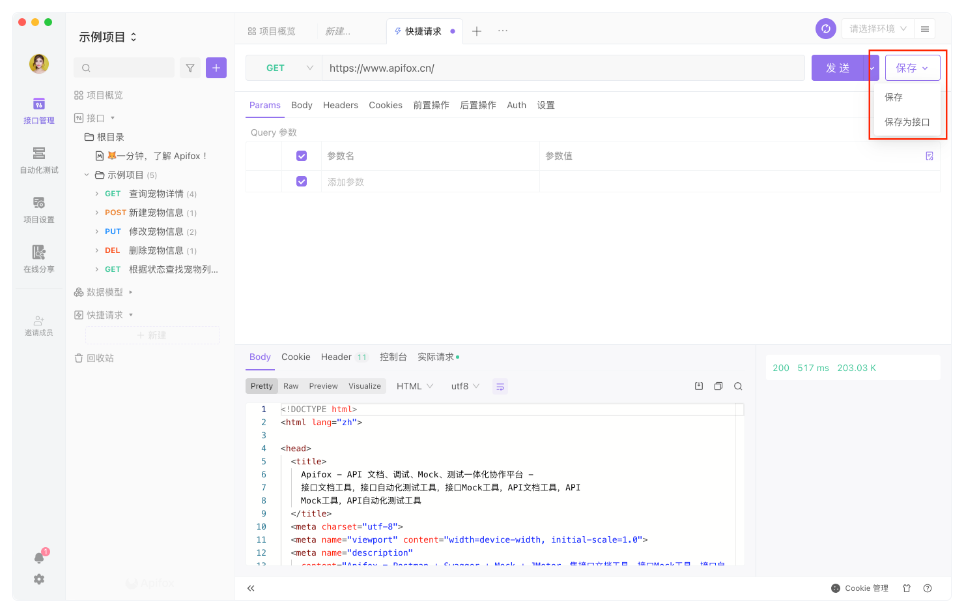


3、输入接口 URL 及参数，即可快速请求接口。

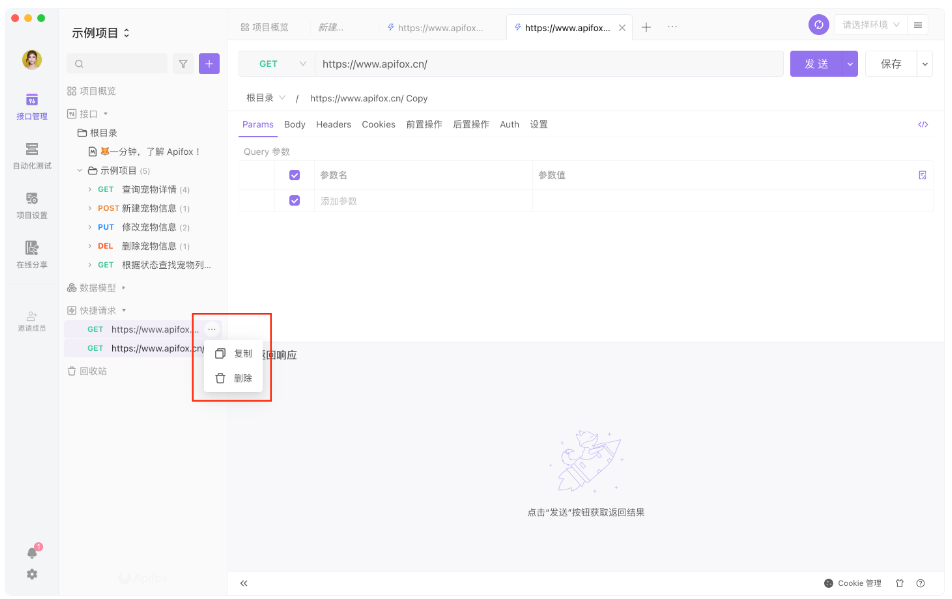


注意：如果输入的 URL 不是以http://或https://起始，实际发出请求的时候会自动加上当前环境里前置 URL。

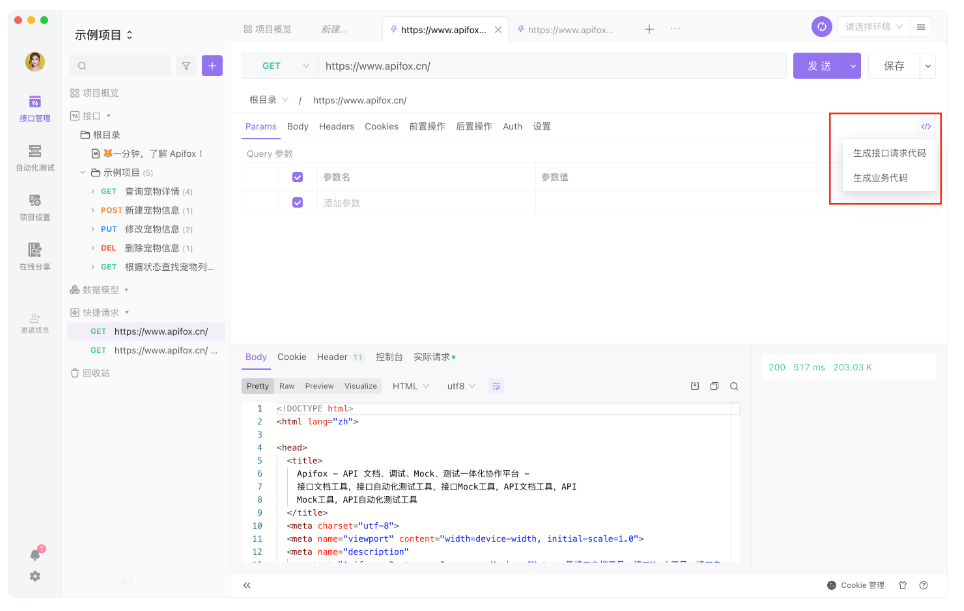
4、调试完成后，可以保存为快捷请求或接口文档



5、在左侧目录可以管理已保存的快捷请求



6、快捷请求 支持 生成代码



## 环境管理

## 环境变量/全局变量/临时变量

## 动态变量/随机参数

## 自定义字段

## 接口唯一标识

## 接口批量管理

## 语言设置

# 核心功能



# 最佳实践

**1、前端**（或**后端**）：在 **Apifox** 上定好接口文档初稿。

**2、前后端**：一起评审、完善接口文档，定好接口用例。

**3、前端**：使用系统根据接口文档自动生成的 Mock 数据进入开发，无需手写 mock 规则。

**4、后端**：使用接口用例调试开发中接口，只要所有接口用例调试通过，接口就开发完成了。如开发过程中接口有变化，调试的时候就自动更新了文档，零成本的保障了接口维护的及时性。

**5、后端**：每次调试完一个功能就保存为一个接口用例。

**6、测试人员**：直接使用接口用例测试接口。

7、所有接口开发完成后，**测试人员**（也可以是**后端**）使用集合测试功能进行多接口集成测试，完整测试整个接口调用流程。

**8、前后端**都开发完，前端从Mock 数据切换到正式数据，联调通常都会非常顺利，因为前后端双方都完全遵守了接口定义的规范。

# 生成业务代码

1、根据接口/模型定义，自动生成各种语言/框架的业务、模型代码。

2、支持 TypeScript、Java、Go、Swift、ObjectiveC、Kotlin、Dart、C++、C#、Rust 等 130 种语言及框架。

3、支持自定义代码模板，自动生成符合自己团队的架构规范的代码，满足各种个性化的需求。

# 数据导入导出

1、支持导出 OpenAPI (Swagger)、Markdown、Html 等数据格式。

2、支持导入 OpenAPI (Swagger)、Postman、HAR、RAP2、JMeter、YApi、Eolinker、RAML、DOClever 、Apizza 、DOCWAY、ShowDoc、I/O Docs、WADL、Google Discovery 等数据格式。

# 功能测试与性能测试的比较

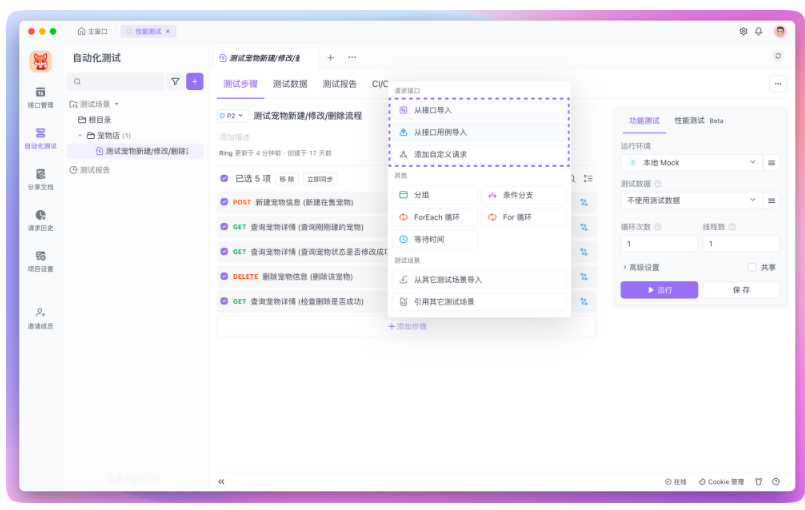


# 性能测试

https://apifox.com/blog/performance-testing/#%E6%AD%A5%E9%AA%A41%EF%BC%9A%E7%BC%96%E6%8E%92%E6%80%A7%E8%83%BD%E6%B5%8B%E8%AF%95%E5%9C%BA%E6%99%AF

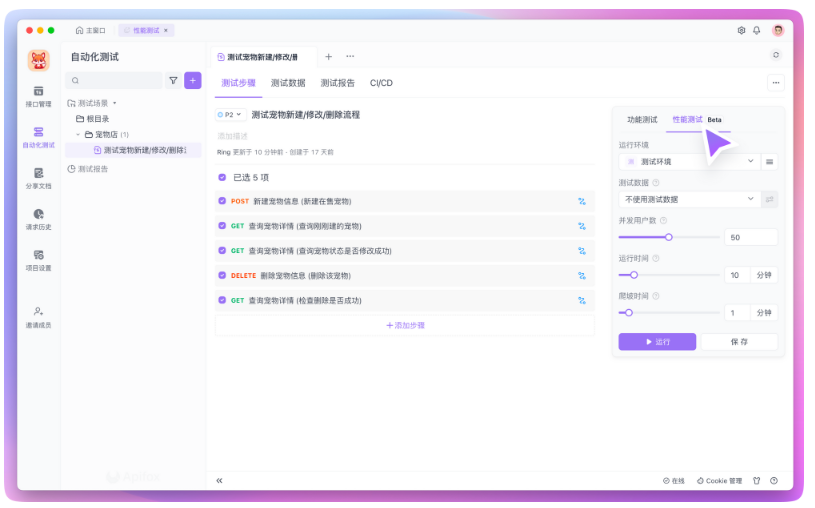
## 1、编排性能测试场景

在启动性能测试前，首先需要**编写测试用例**，可以从接口或接口用例导入，也可以自定义请求。当测试步骤准备完毕，建议先执行功能测试以确保接口按预期工作，然后再进行性能测试。



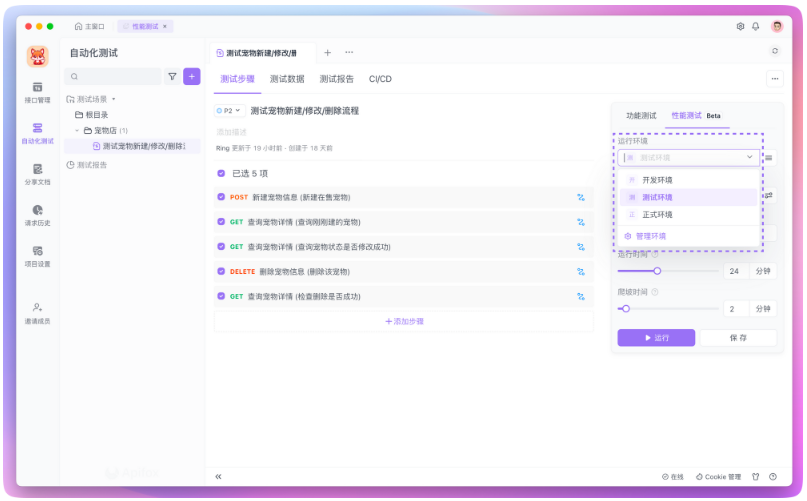
## 2、设置性能测试运行配置

在「自动化测试」功能界面，你会找到一个专为「性能测试」设计的模块。在这个模块里，你可以根据需要设置各种配置项，为即将执行的性能测试做好准备。



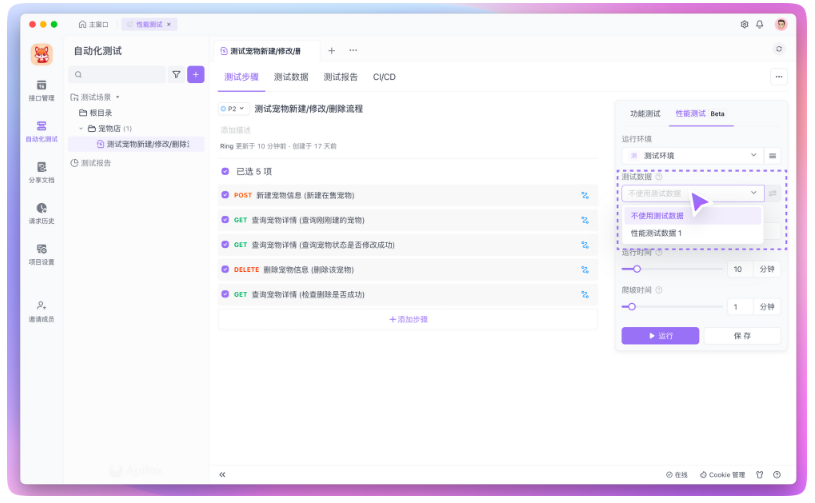
### 运行环境

指定「性能测试」运行期间请求的后台服务环境，**默认自动采用当前项目设定的环境配置**。

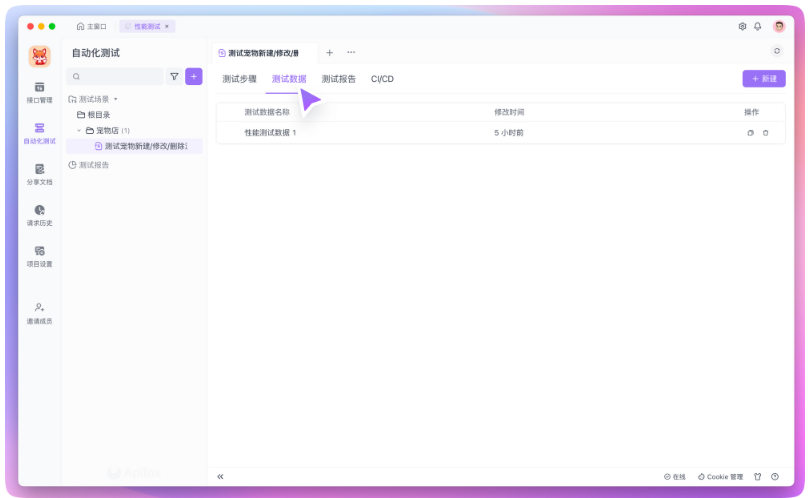


### 测试数据（可选）

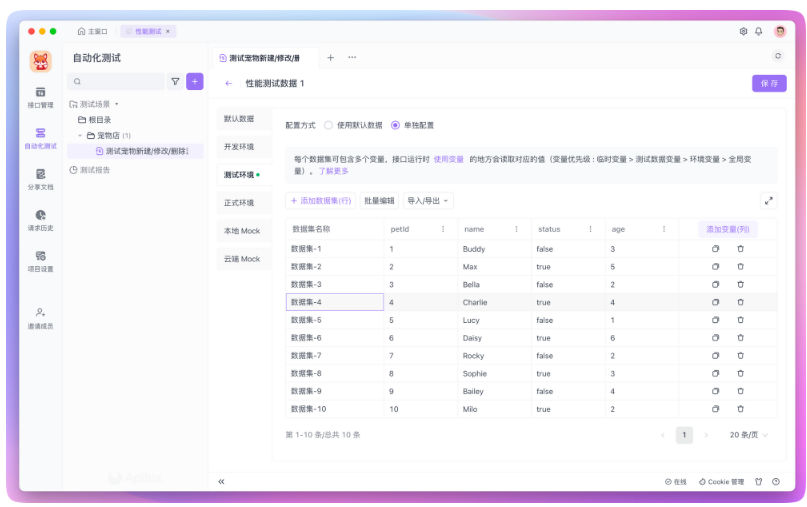
**测试数据指的是性能测试过程中用来发起请求的数据集**，通过使用测试数据，让每个并发用户能够使用不同的参数进行请求，用以更好的模拟现实世界的流量。



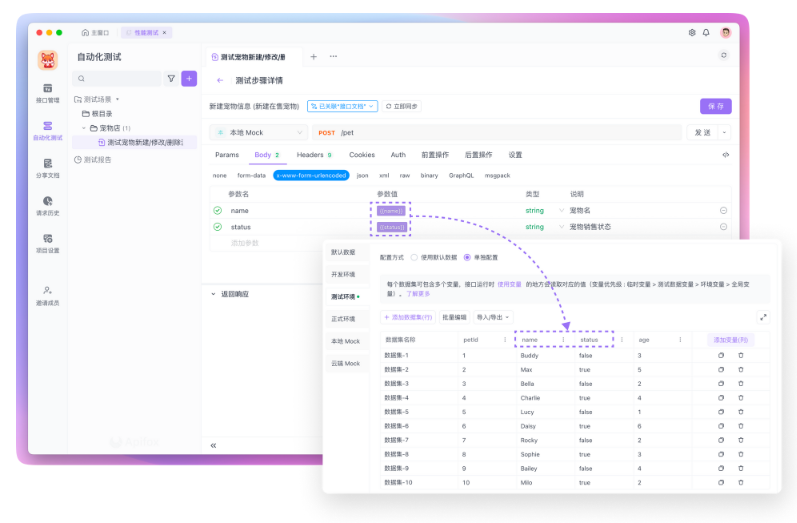
如果你要使用测试数据，你可以在界面上点击「测试数据」标签，然后新建一个测试数据。



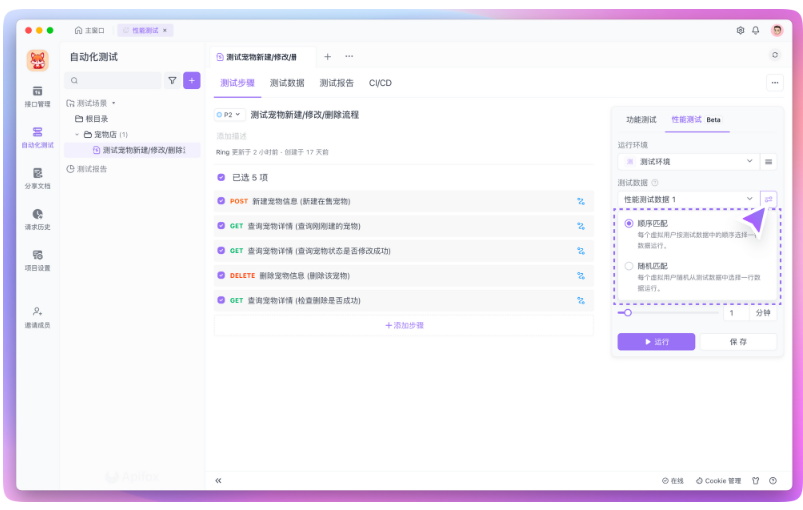
接着在相应的环境中配置数据集，你可以**手动添加**，也可以**通过 JSON 或 CSV 格式导入**，编辑完毕后保存即可。



最后，在测试步骤详情页中通过变量与测试数据关联。



若你在性能测试中选择关联测试数据，那么并发用户将使用测试数据中定义的变量进行请求，你可以**选择「随机匹配」或者「顺序匹配」这两种模式进行测试**。



「随机匹配」模式：在此模式下，每个并发用户将从测试数据中随机选择一行数据进行操作。这确保了每个并发用户都能获取到一条测试数据，从而保证性能测试的广覆盖性。

「顺序匹配」模式：此模式下，每个并发用户将按照测试数据的顺序选取下一行数据进行操作。需要注意的是，如果执行测试的并发用户数超过了提供的测试数据行数，那么超出部分的并发用户将无法参与测试。因此，在使用此模式时，请确保你有足够多的测试数据以支持所有并发用户的测试需求。

### 并发用户数（虚拟用户数）

目前性能测试最多支持同时模拟 100 名并发用户，这些并发用户会在你指定的测试时间内，并行地模拟真实在线用户行为，以重复执行测试场景。通过这种方式，你可以有效地模拟高并发场景，以评估 API 在面对大量用户请求时的性能和稳定性。

### 运行时间

运行时间定义了性能测试的总时长，即在此设定时间内，每个并发用户将持续不断地循环执行测试场景中定义的所有接口请求。当前，性能测试支持的最长测试时长为 60 分钟。

### 爬坡时间

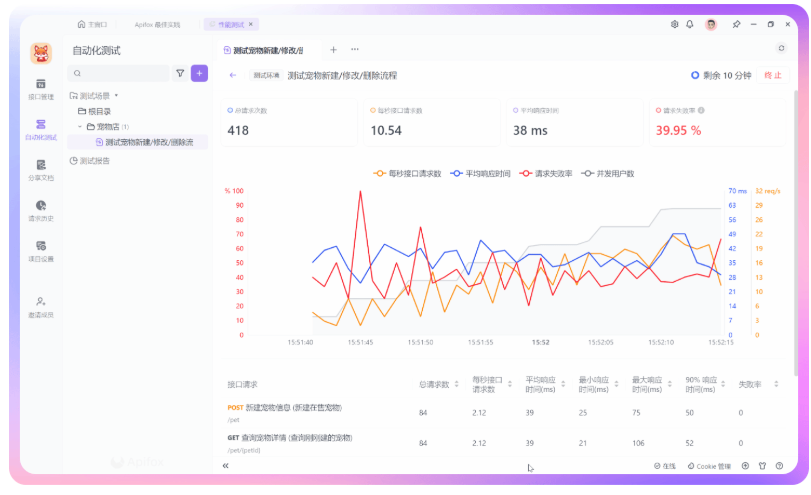
在一些情况下，大量用户并非瞬时访问服务，而是随着时间推移，请求越来越多。爬坡时间就是用于模拟用户访问量随时间增长的情况，即在测试的初期并不是立即使用所有并发用户，而是在设定的 X 分钟内（X 为你设定的时间），用户数量会线性增加直到达到所设定的并发用户总数。如果将 X 值设置为 0，意味着测试一开始就会启用全部并发用户进行压力测试。

以上的「可配置项」配置完成后，保存设置，然后点击「运行」按钮，即可开始执行性能测试。

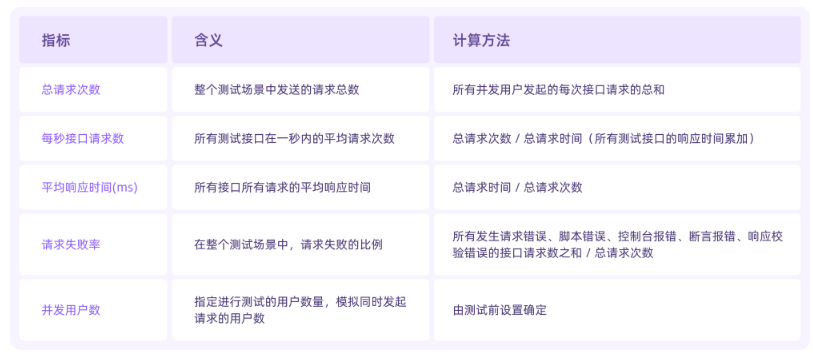
## 3、运行性能测试

性能测点击「运行」后，性能测试就会启动。**Apifox 的性能测试是从用户电脑（而不是 Apifox 服务器）发起的**，会使用当前电脑的硬件资源来执行你已经配置好的测试步骤、测试数据和并发请求。

启动性能测试后，Apifox 将会提供一个直观的可视化面板，用以实时监测，并展现关键性能指标，包括**各个接口的总请求数、每秒接口请求数、平均响应时间、最大/最小响应时间**，以及**请求的失败率**等信息。

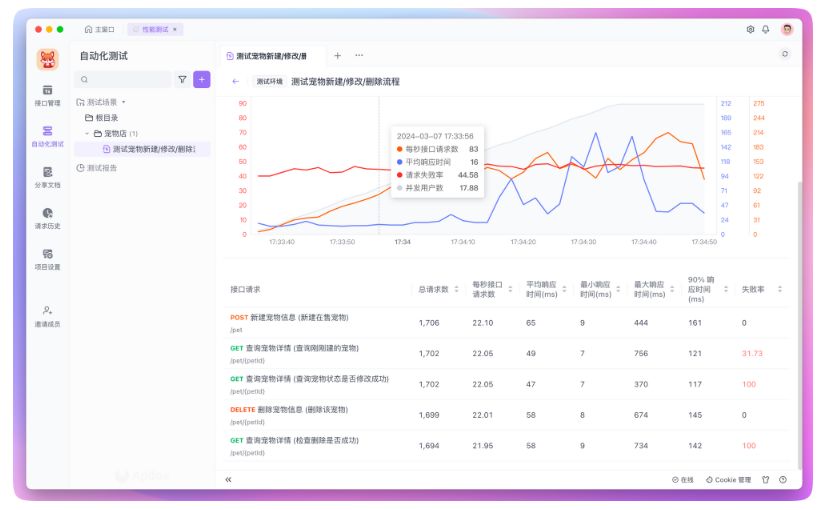


### 综合数据指标



### 各个接口的具体数据指标

在可视化面板下方，列出了测试场景中各个接口的具体数据指标，便于你根据需要进行查看和分析。



### ****单个接口的性能测试指标****

