

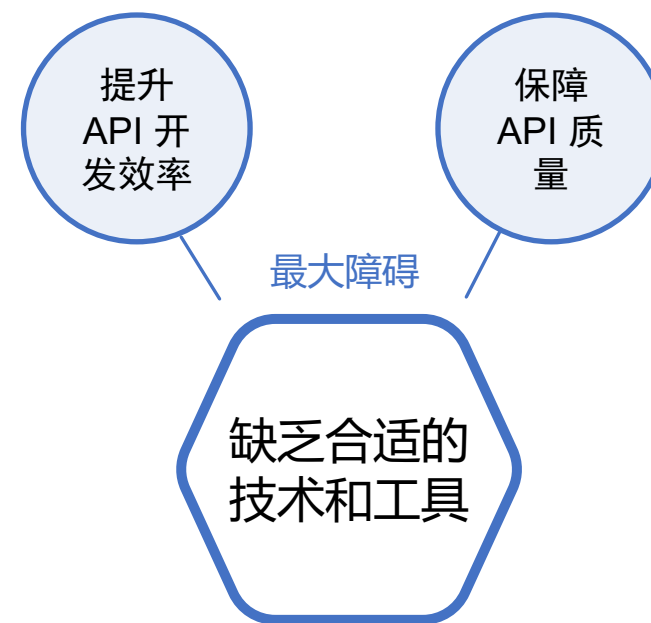
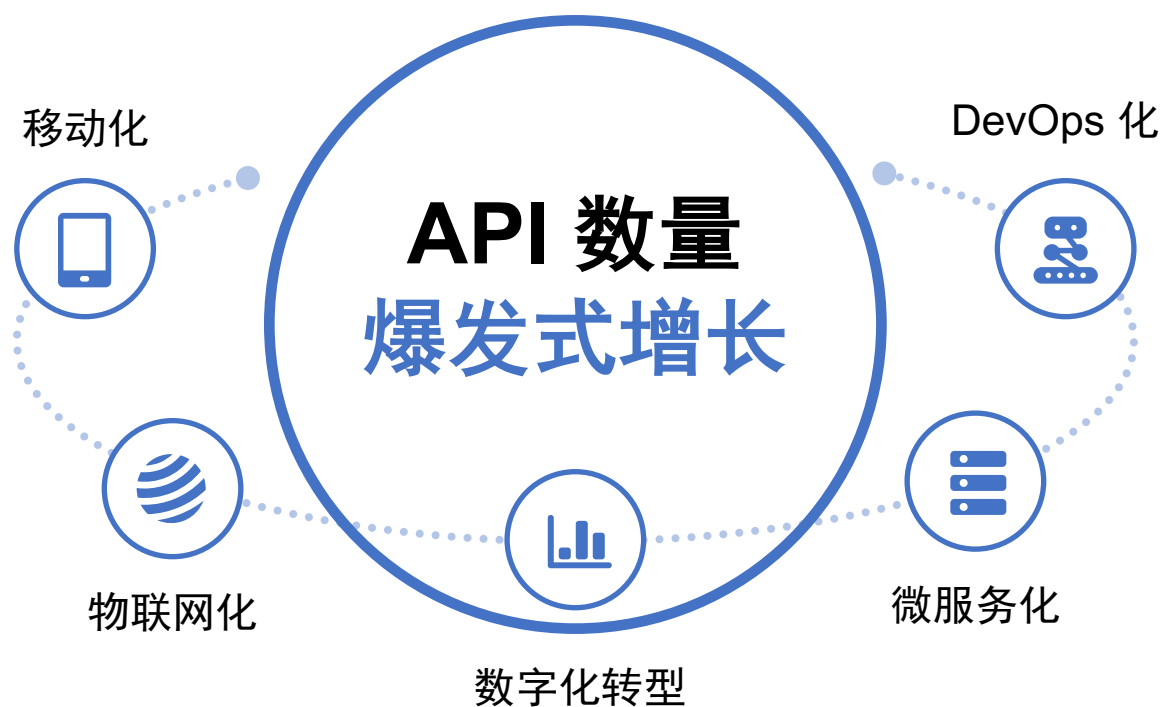


# API 设计、开发、测试 一体化协作平台

广州睿狐科技有限公司  
Apifox Inc.

Apifox.cn  
节省研发团队的每一分钟

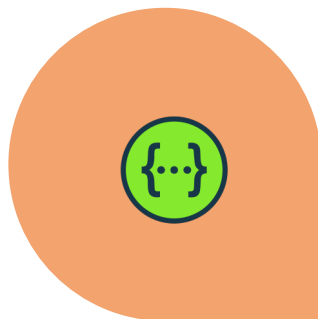
# 行业情况



# 常用解决方案

API 设计者

Swagger  
API 文档设计

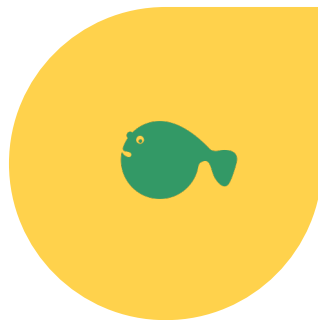


后端开发

Postman  
API 开发调试



Mock.js  
API 数据Mock



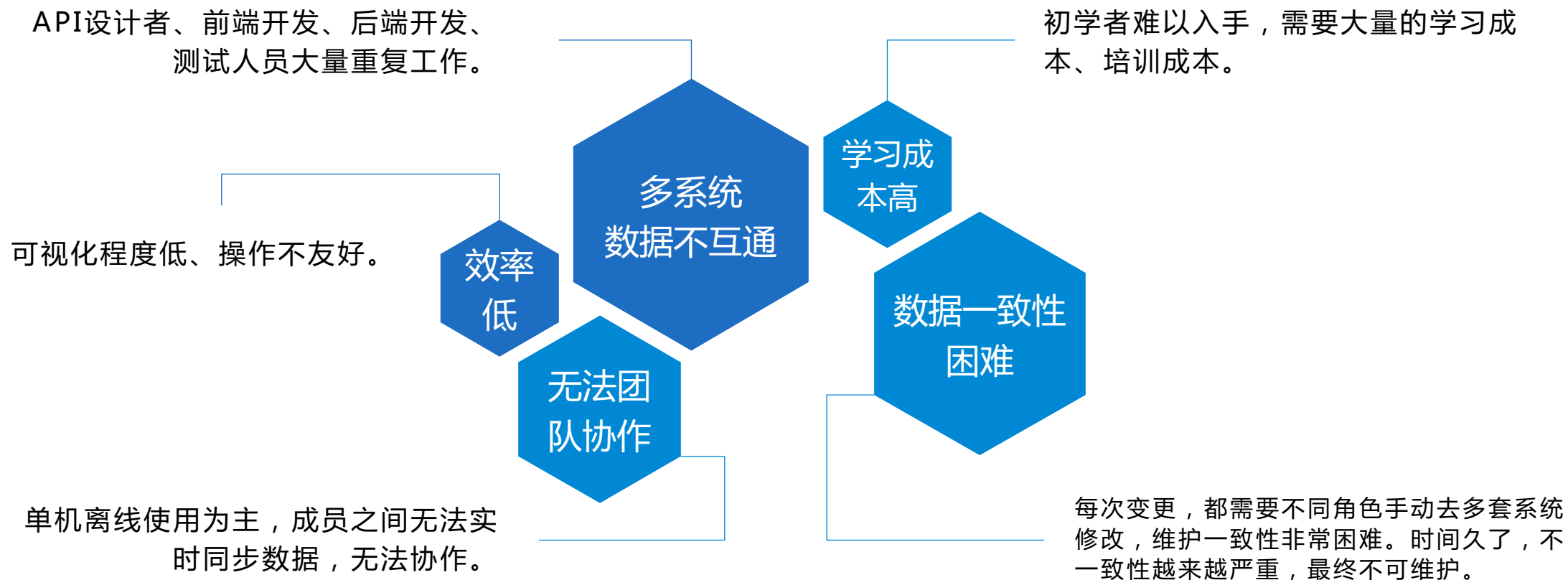
JMeter  
API 自动化测试  
API 压力测试



前端开发

测试人员

# 存在问题



# Apifox

= Postman + Swagger + Mock + JMeter

宠物商店

公开

接口管理

自动化测试

项目设置

工具箱

在线分享

邀请成员

更多

数据模型

宠物相关

Pet

Category

Tag

宠物商店

公开

接口管理

自动化测试

项目设置

工具箱

在线分享

邀请成员

更多

数据模型

宠物相关

Pet

Category

Tag

项目概览

POST 新建宠物资料

GET 创建项目

GET 查询宠物详情

+

...

文档

修改文档

运行

高级 Mock

基础信息

GET

/rootapi

已发布

创建时间：2022年2月24日

修改时间：2个月前

最后修改人：狐哥

创建人：狐哥

责任人：狐哥

分组：宠物

请求参数

参数名	位置	类型	必填	说明
petId	path	string	否	宠物 ID

返回响应

成功 (200)

ID 不存在 (404)

记录不存在 (404)

HTTP 状态码：200

内容格式：JSON

object {2}

id	integer	必需	宠物 ID
category	object {2}	必需	分类
name	string	必需	宠物名称 示例值
photoUrls	array [string]	必需	照片 URL

API 自动化测试

864

已完成

通过 30% 258

失败 60% 520

未测 10% 86

API Mock

本地 Mock

云端 Mock

# 核心功能

API 设计者

## API 文档

比 Swagger、Markdown 更好用

后端开发

## API 调试

比 Postman 更强大

前端开发

## API 数据 Mock

比 Mock.js 更智能

测试人员

## API 自动化测试

比 JMeter 更高效

# 更多特性



## CI 持续集成

1. 支持命令行方式运行 API 测试 (Apifox CLI)。
2. 支持集成 Jenkins 等持续集成工具。



## 数据库操作

1. 支持读取数据库数据，作为 API 请求参数使用。
2. 支持读取数据库数据，用来校验(断言) API 请求是否成功。



## 团队协作

1. 接口数据云端同步，实时更新。
2. 成熟的团队/项目权限管理，支持管理员、普通成员、只读成员等角色设置，满足各类企业的需求。



## 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、NEI、RAML、DOClever、Apizza、DOCWAY、ShowDoc、apiDoc、I/O Docs、WADL、Google Discovery 等数据格式。



## 支持 HTTP、TCP、RPC

1. 支持 HTTP(s) 接口管理。
2. 支持 Socket (TCP) 接口管理。
3. 后续将会支持 GraphQL、Dubbo、gRPC、WebSocket 等协议接口。



## 自动生成代码

1. 根据接口/模型定义，自动生成各种语言/框架的业务代码和 API 请求代码。
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。

# 解决的问题

1. 一套系统、一份数据，解决多个系统之间的数据同步问题。
2. 只要定义好接口文档，接口调试、数据 Mock、接口测试即可直接使用，无需再次定义。
3. 接口文档和接口开发调试使用同一个工具，接口调试完成后即可保证和接口文档定义完全一致。
4. 高效、及时、准确！

# 最佳实践

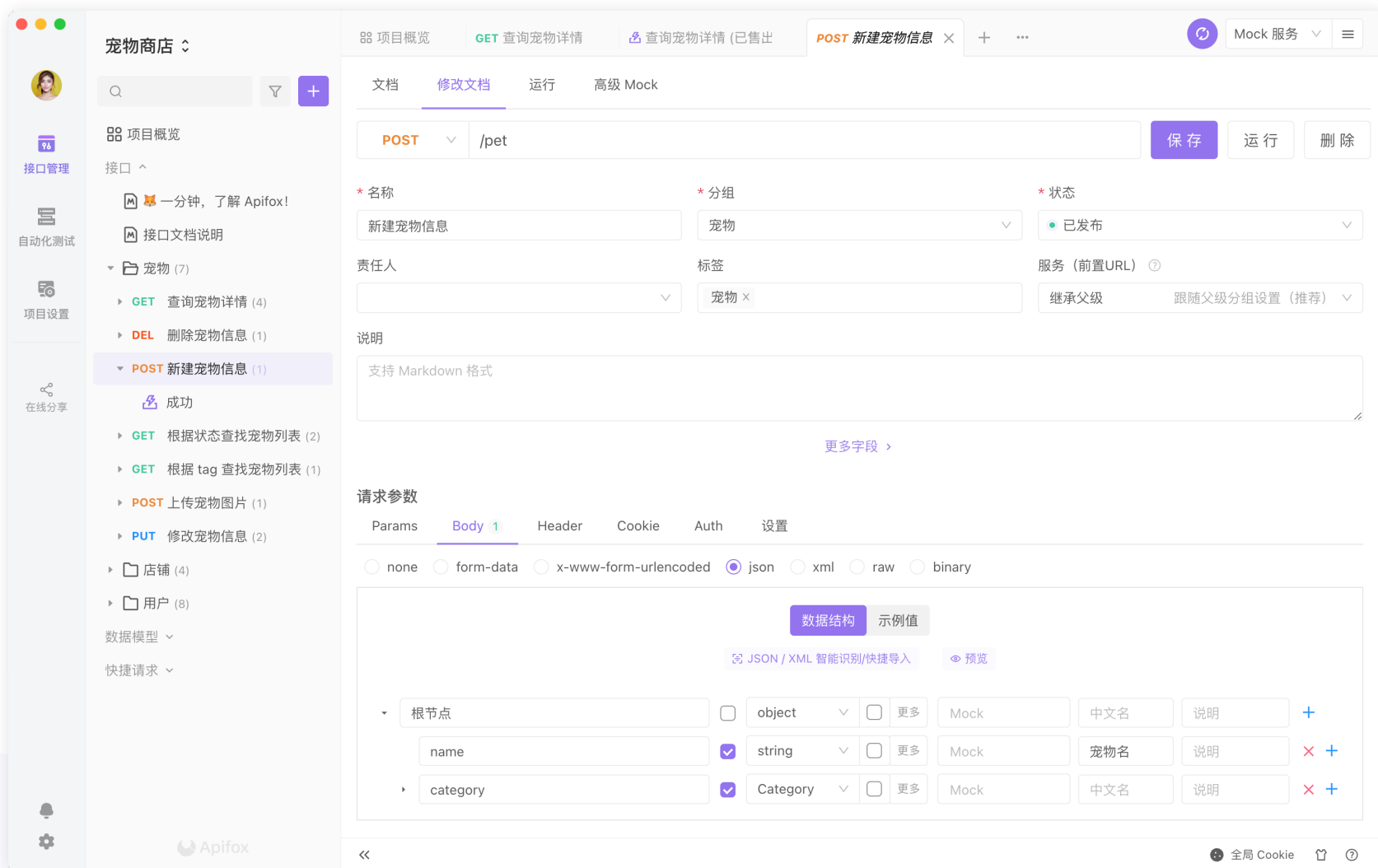
1. **前端（或后端）**：在 **Apifox** 上定好接口文档初稿。
2. **前后端**：一起评审、完善接口文档，定好接口用例。
3. **前端**：使用系统根据接口文档自动生成的 Mock 数据进入开发，无需手写 mock 规则。
4. **后端**：使用接口用例 调试开发中接口，只要所有接口用例调试通过，接口就开发完成了。如开发过中接口有变化，调试的时候就自动更新了文档，零成本的保障了接口维护的及时性。
5. **后端**：每次调试完一个功能就保存为一个接口用例。
6. **测试人员**：直接使用接口用例测试接口。
7. 所有接口开发完成后，**测试人员**（也可以是**后端**）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。

# 谁在使用 Apifox



# 接口文档设计

# 接口文档设计



宠物商店

项目概况

接口

一分钟，了解 Apifox!

接口文档说明

宠物 (7)

GET 查询宠物详情 (4)

DEL 删除宠物信息 (1)

POST 新建宠物信息 (1)

成功

GET 根据状态查找宠物列表 (2)

GET 根据 tag 查找宠物列表 (1)

POST 上传宠物图片 (1)

PUT 修改宠物信息 (2)

店铺 (4)

用户 (8)

数据模型

快捷请求

项目概况

GET 查询宠物详情

查询宠物详情 (已售出)

POST 新建宠物信息

文档

修改文档

运行

高级 Mock

POST /pet

保存

运行

删除

\* 名称

新建宠物信息

\* 分组

宠物

\* 状态

已发布

责任人

宠物

标签

宠物

服务 (前置URL)

继承父级

跟随父级分组设置 (推荐)

说明

支持 Markdown 格式

更多字段

请求参数

Params

Body

Header

Cookie

Auth

设置

none

form-data

x-www-form-urlencoded

json

xml

raw

binary

数据结构

示例值

JSON / XML 智能识别/快捷导入

预览

根节点

object

name

string

category

Category

Mock

中文名

说明

宠物名

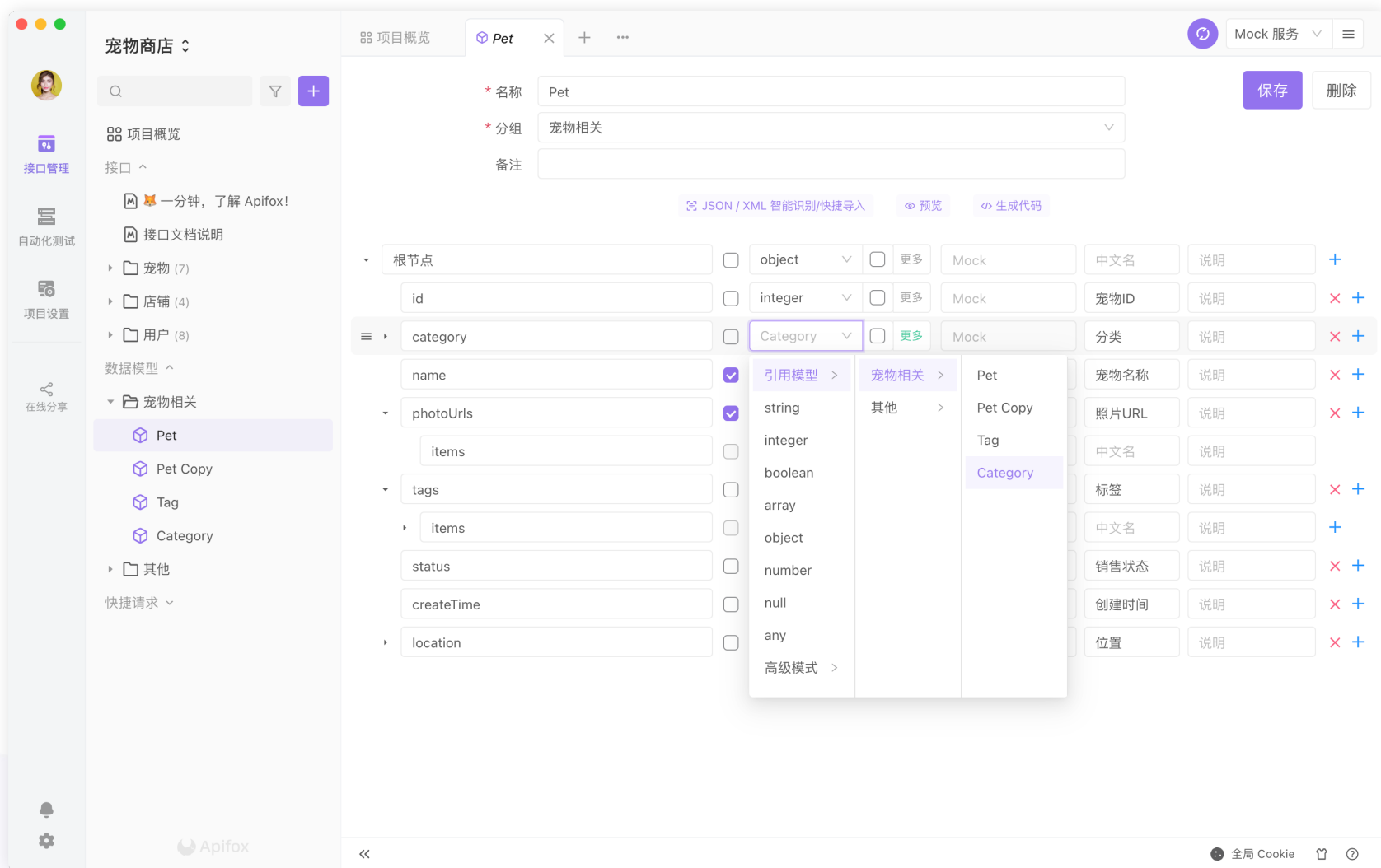
说明

说明

全局 Cookie

1. 完全可视化
2. 零学习成本
3. 遵循 OpenAPI 规范

# 数据模型



The screenshot displays the Apifox web application interface for managing a data model. The left sidebar contains navigation options: 宠物商店 (Pet Store), 接口管理 (API Management), 自动化测试 (Automated Testing), 项目设置 (Project Settings), and 在线分享 (Online Sharing). The main workspace is titled '宠物商店' and shows a project overview for 'Pet'. The '数据模型' (Data Model) section is active, displaying a tree structure of the model. The 'Pet' model is expanded, showing fields: 'id' (integer), 'category' (reference to 'Category'), 'name' (string), 'photoUrls' (array of strings), 'tags' (array of strings), 'status' (string), 'createTime' (string), and 'location' (string). A dropdown menu is open for the 'category' field, showing options: '引用模型' (Reference Model) with '宠物相关' (Pet Related) and '其他' (Other), and '高级模式' (Advanced Mode). The '宠物相关' option is selected, showing a list of models: 'Pet', 'Pet Copy', 'Tag', and 'Category'. The 'Category' model is highlighted. The right sidebar contains a 'Mock 服务' (Mock Service) section with '保存' (Save) and '删除' (Delete) buttons. The bottom of the interface shows the Apifox logo and a '全局 Cookie' (Global Cookie) button.

1. 完全可视化
2. 支持模型之间嵌套引用
3. 支持 JSON/XML 智能导入
4. 遵循 JSON Schema 规范
5. 支持 oneOf、allOf 等高级组合模式

# 接口文档

宠物商店

接口管理

自动化测试

项目设置

在线分享

项目概览

接口

宠物 (7)

GET 查询宠物详情 (4)

DEL 删除宠物信息 (1)

POST 新建宠物信息 (1)

成功

GET 根据状态查找宠物列表 (2)

GET 根据 tag 查找宠物列表 (1)

POST 上传宠物图片 (1)

PUT 修改宠物信息 (2)

店铺 (4)

用户 (8)

数据模型

快捷请求

GET 查询宠物详情

查询宠物详情 (已售出)

POST 新建宠物信息

Mock 服务

文档

修改文档

运行

高级 Mock

新建宠物信息 #11221575

运行

生成代码

删除

POST /pet

已发布

宠物

创建时间: 2022年2月9日

修改时间: 2个月前

最后修改人: apifox111

创建人: apifox111

责任人: 未设置

分组: 宠物

MOCK

名称	来源	URL / 参数	操作
成功 (200)	接口响应	http://127.0.0.1:4523/mock/622362/pet	快捷请求

请求参数

Body 参数(application/json)

数据结构

示例值

```

object {2}
  name string 宠物名
  category object {2}
    id integer 分类ID int64
    name string 分类名

```

返回响应

成功 (200)

宠物商店

一分钟, 了解 Apifox!

接口文档说明

宠物

GET 查询宠物详情

DEL 删除宠物信息

POST 新建宠物信息

GET 根据状态查找宠物...

GET 根据 tag 查找宠物...

POST 上传宠物图片

PUT 修改宠物信息

店铺

用户

查询宠物详情

GET https://www.apifox.cn//pet/{petId}

修改时间: 2 分钟前 责任人: 未设置

请求参数

参数名	位置	参数类型	必填	说明
petId	path	text	是	宠物 ID

返回响应

成功 (200) ID 不正确 (400) 记录不存在 (404)

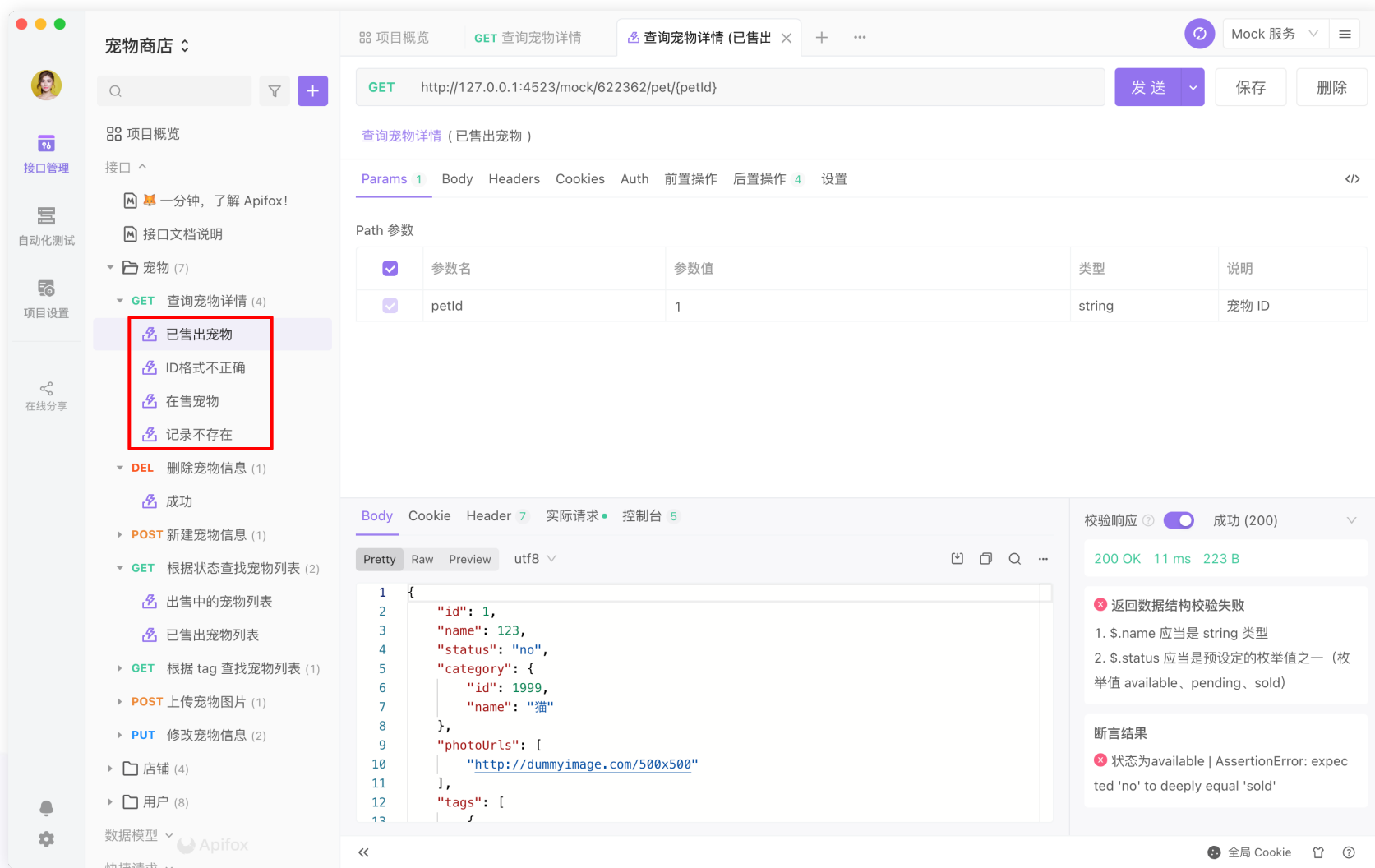
HTTP 状态码: 200 内容格式: JSON

object {8} Pet XML

id	integer	宠物ID	int64
category	object {2}	Category	XML
id	integer	分类ID	int64
name	string	分类名	
name	string	宠物名称	示例值
photoUrls	array[string]	照片URL	XML
tags	array[object] {2}	Tag	XML
id	integer		int64
name	string	标签名	
status	string	销售状态	枚举(3)
createTime	string	创建时间	

# 接口用例/接口调试

# 接口用例



宠物商店

GET 查询宠物详情

查询宠物详情 (已售出宠物)

GET http://127.0.0.1:4523/mock/622362/pet/{petId}

发送 保存 删除

查询宠物详情 (已售出宠物)

Params 1 Body Headers Cookies Auth 前置操作 后置操作 4 设置

Path 参数

参数名	参数值	类型	说明
petId	1	string	宠物 ID

Body

```

1 {
2   "id": 1,
3   "name": 123,
4   "status": "no",
5   "category": {
6     "id": 1999,
7     "name": "猫"
8   },
9   "photoUrls": [
10    "http://dummyimage.com/500x500"
11  ],
12  "tags": [
13  ]
14 }

```

校验响应 成功 (200)

200 OK 11 ms 223 B

返回数据结构校验失败

- \$.name 应当是 string 类型
- \$.status 应当是预设定的枚举值之一 (枚举值 available、pending、sold)

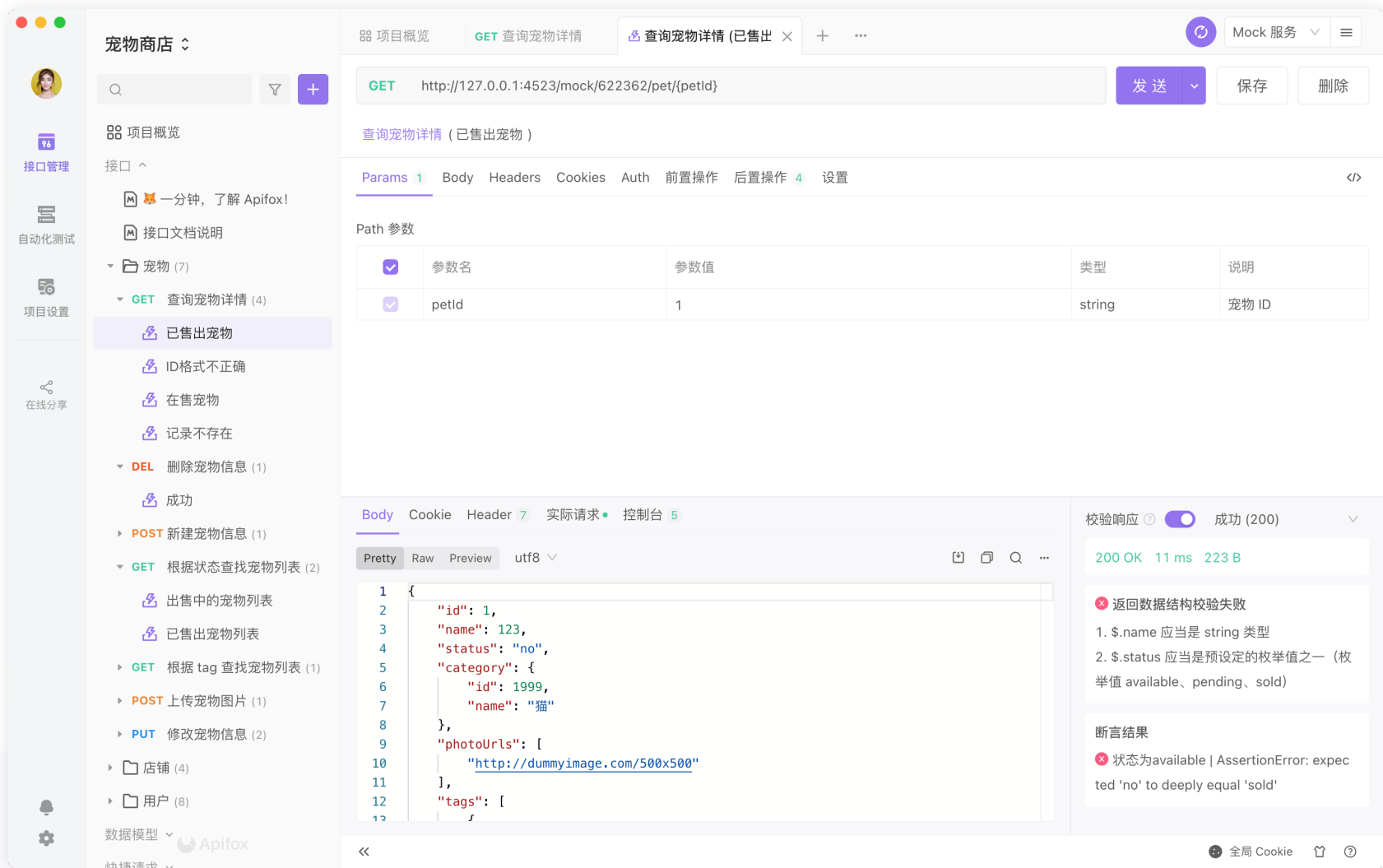
断言结果

状态为available | AssertionError: expected 'no' to deeply equal 'sold'

1. 一个接口多个用例

2. 自动跟随接口变更

# 接口调试



宠物商店

GET 查询宠物详情

查询宠物详情 (已售出)

GET http://127.0.0.1:4523/mock/622362/pet/{petId}

发送 保存 删除

查询宠物详情 (已售出宠物)

Params 1 Body Headers Cookies Auth 前置操作 后置操作 4 设置

Path 参数

参数名	参数值	类型	说明
petId	1	string	宠物 ID

Body Cookie Header 7 实际请求 控制台 5

Pretty Raw Preview utf8

```
1 {
2   "id": 1,
3   "name": 123,
4   "status": "no",
5   "category": {
6     "id": 1999,
7     "name": "猫"
8   },
9   "photoUrls": [
10    "http://dummyimage.com/500x500"
11  ],
12  "tags": [
13  ]
14 }
```

校验响应 成功 (200)

200 OK 11 ms 223 B

返回数据结构校验失败

- \$.name 应当是 string 类型
- \$.status 应当是预设定的枚举值之一 (枚举值 available、pending、sold)

断言结果

状态为available | AssertionError: expected 'no' to deeply equal 'sold'

Postman 有的功能  
Apifox 基本都有

环境变量、全局变量、前置/后置脚本、Cookie/Session 全局共享等...

# 环境变量/全局参数

全局

全局变量

全局参数

环境

测试环境

正式环境

Mock 服务

+ 新建环境

正式环境

\* 环境名称 正式环境 共用

服务 (前置URL)

服务名	前置 URL
默认服务 默认	https://www.apifox.cn/

+ 添加服务

环境变量

变量名	远程值	本地值
token		
添加变量		

关闭 保存

全局

全局变量

全局参数

环境

测试环境

正式环境

Mock 服务

+ 新建环境

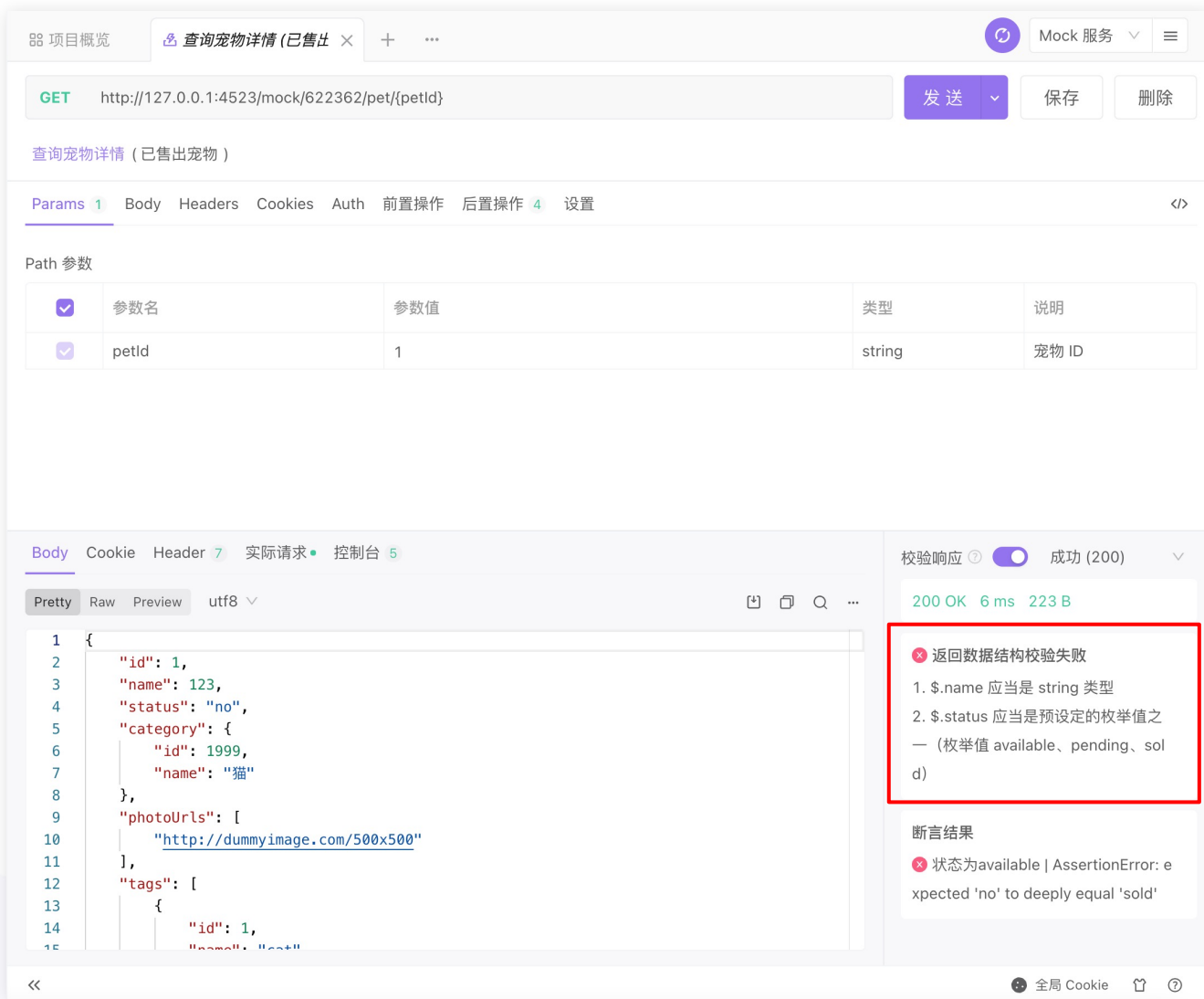
全局参数

Header 1 Cookie Query Body

参数名	类型	必填	默认值	默认启用	说明
X-Client-Versic	string	<input checked="" type="checkbox"/>	1.0.0	<input checked="" type="checkbox"/>	客户端版本号
	string	<input type="checkbox"/>		<input checked="" type="checkbox"/>	

关闭 保存

# 自动校验接口数据



项目概览 查询宠物详情 (已售出) × + ... Mock 服务

GET http://127.0.0.1:4523/mock/622362/pet/{petId} 发送 保存 删除

查询宠物详情 (已售出宠物)

Params 1 Body Headers Cookies Auth 前置操作 后置操作 4 设置

Path 参数

参数名	参数值	类型	说明
petId	1	string	宠物 ID

Body Cookie Header 7 实际请求 控制台 5

Pretty Raw Preview utf8

```
1 {
2   "id": 1,
3   "name": 123,
4   "status": "no",
5   "category": {
6     "id": 1999,
7     "name": "猫"
8   },
9   "photoUrls": [
10    "http://dummyimage.com/500x500"
11  ],
12  "tags": [
13    {
14      "id": 1,
15      "name": "cat"
16    }
17  ]
18 }
```

校验响应 成功 (200) 200 OK 6 ms 223 B

✖ 返回数据结构校验失败

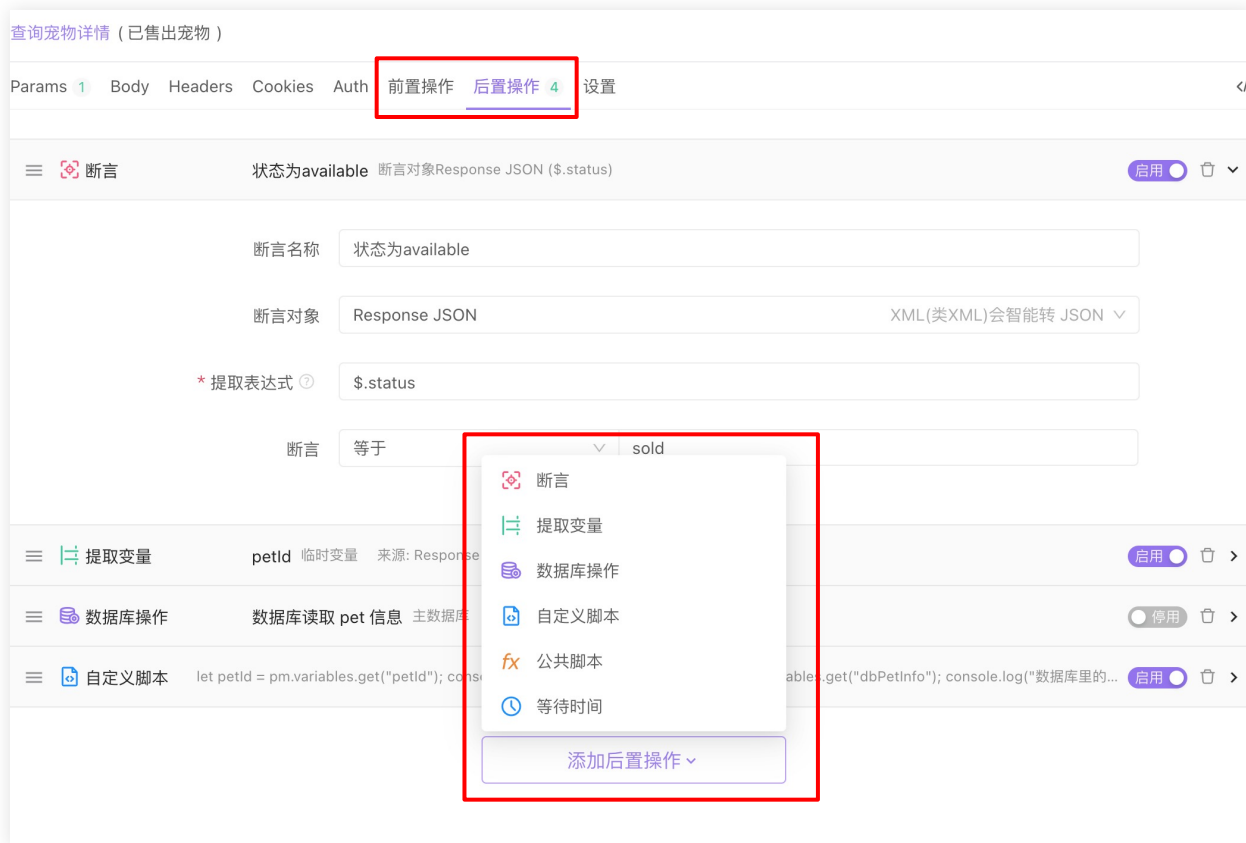
- 1. \$.name 应当是 string 类型
- 2. \$.status 应当是预设定的枚举值之一 (枚举值 available、pending、sold)

断言结果

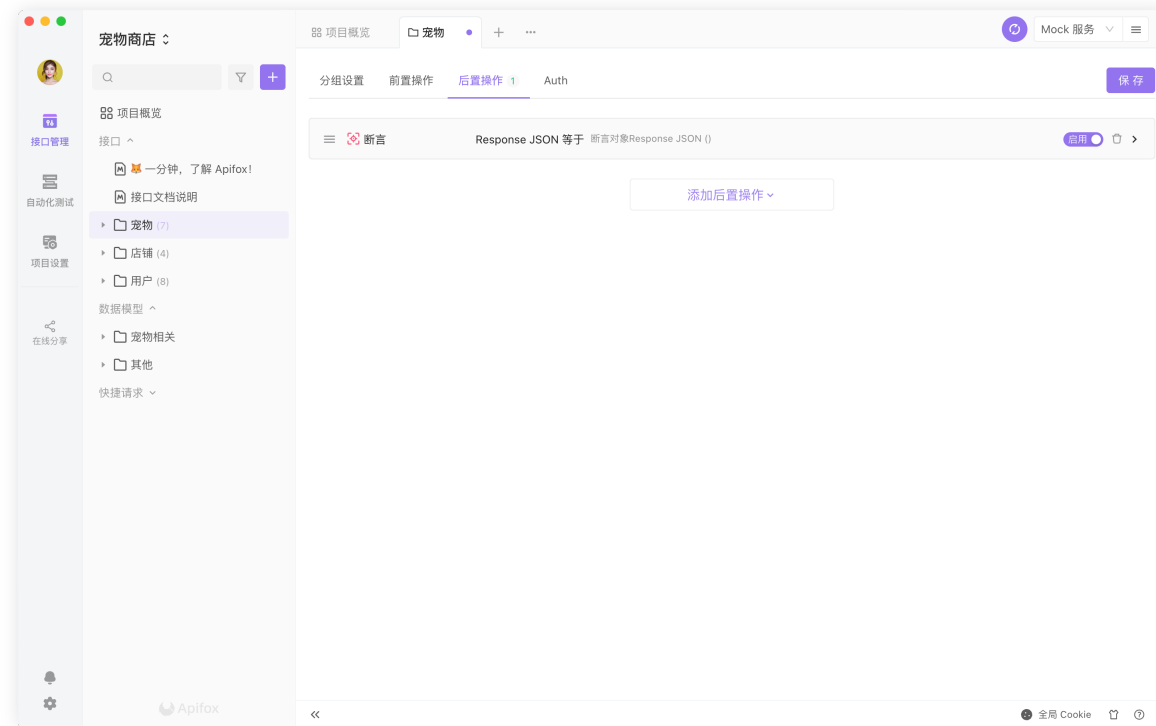
✖ 状态为available | AssertionError: expected 'no' to deeply equal 'sold'

1. 根据数据结构自动校验
2. 完整的 JSON Schema 校验

# 前置操作/后置操作



针对单个接口



针对整个分组

Body Cookie Header 7 实际请求 • 控制台

Pretty Raw Preview utf8 ▾

```
1 {
2   "name": "林养七需强育引",
3   "photoUrls": [
4     "http://dummyimage.com/400x400",
5     "http://dummyimage.com/400x400"
6   ],
7   "tags": [
8     {
9       "name": "持便并张号",
10      "id": 97
11    }
12  ],
13  "createTime": "1983-09-15 18:53:31",
14  "status": "available",
15  "category": {
16    "name": "龙对治边米值革",
17    "id": 37
18  },
19  "id": 69,
20  "location": {
```

校验响应  成功 (200) 

200 OK 14 ms 462 B

✔ 返回数据结构校验通过！

断言结果

❌ 状态为available | AssertionError: expected 'available' to deeply equal 'sold'

# 提取变量

GET http://127.0.0.1:4523/mock/622849/pet/{petId} 发送 保存 删除

查询宠物详情 (已售出宠物)

Params 1 Body Headers Cookies Auth 前置操作 后置操作 4 设置 </>

≡ 断言 状态为available 断言对象Response JSON (\$..status) 启用 🗑 >

≡ 提取变量 petId 临时变量 来源: Response JSON (\$..id) 启用 🗑 ▼

\* 变量名称

petId

变量类型

临时变量

提取来源

Response JSON

XML(类XML)会智能转 JSON ▼

提取范围

☐ 整个返回数据 ☒ 提取部分

\* 提取表达式 ?

\$.id

Body Cookie Header 7 实际请求 控制台 7

"已设置临时变量【petId】，值为【1】"

校验响应 ? 成功 (200) ▼

200 OK 6 ms 223 B

1. 可视化

2. JSON Path 提取

# 数据库操作

GET http://127.0.0.1:4523/mock/622849/pet/{petId} 发送 保存 删除

查询宠物详情 (已售出宠物)

数据库操作 数据库读取 pet 信息 主数据库 SELECT \* FROM pets WHERE id={{petId}} 启用

\* 操作名称 数据库读取 pet 信息

\* 数据库连接 主数据库

\* SQL 命令 SELECT \* FROM pets WHERE id={{petId}}

控制台打印结果 ☒

提取结果到变量

变量名	变量类型	JSON Path 表达式
dbPetInfo	临时变量	\$.[0]
变量名	请选择	如: \$.[0].name, 默认为: \$

注意: 查询 SQL 语句返回结果是 数组 格式, 如:

Body Cookie Header 7 实际请求 控制台 7

"数据库运行结果"

```
[
  {
    "id": "1"
    "name": "kitty"
    "status": "available"
  }
]
```

校验响应 成功 (200) 200 OK 6 ms 223 B

返回数据结构校验失败

- 1. \$.name 应当是 string 类型
- 2. \$.status 应当是预设定的枚举值之一 (枚举值 available、pending、sold)

1. 读取数据库数据

2. 写入数据库数据

# 自定义脚本

Params 2 Body Headers 1 Cookies 1 Auth 前置操作 后置操作 1 设置

自定义脚本 console.log("调用外部 python 代码, 进行md5加密"); try { const pyResult = fox.execute('testmd5.py',[123456]); co... 启用

```
1
2 console.log("调用外部 python 代码, 进行md5加密");
3
4 try {
5     const pyResult = fox.execute('testmd5.py',[123456]);
6     console.log('加密前参数值', 123456);
7     console.log('MD5加密后', pyResult);
8 } catch (e) {
9     console.error(e.message);
10 }
11
```

代码片段:

- 获取环境变量
- 设置环境变量
- 获取临时变量
- 设置临时变量
- 请求接口
- Response 状态码为 200
- Response Body 包含字符串

Body Cookie Header 7 实际请求 控制台 3

"调用外部 python 代码, 进行md5加密"

"加密前参数值"	123456
"MD5加密后"	"e10adc3949ba59abbe56e057f20f883e"

校验响应 成功 (200)

200 OK 24 ms 119 B

1. 语法 100% 兼容 Postman

2. 支持运行其他任何语言代码

javascript、java、python、php、js、  
BeanShell、go、shell、ruby、lua ...

# Mock 数据

# 零配置 Mock 接口数据

```
{
  "username": "黎勇",
  "phone": "13247278136",
  "age": 23,
  "avatar": "http://dummyimage.com/100x100",
  "description": "广高六被严",
  "location": {
    "province": "江西省",
    "city": "丽江市",
    "address": "安徽省日喀则地区石狮市"
  },
  "registerTime": 72567224950,
  "createAt": "1998-07-24 04:05:01",
  "registerIp": "83.18.192.180"
}
```

Apifox

```
{
  "username": "Lorem in in esse",
  "phone": "anim",
  "age": 65335358,
  "avatar": "deserunt aute pariatur sed",
  "description": "in",
  "location": {
    "province": "enim",
    "city": "Duis non dolor",
    "address": "deserunt cillum consecter"
  },
  "registerTime": 94673859,
  "createAt": "in aute magna",
  "registerIp": "culpa enim"
}
```

同类工具

# 智能 Mock 数据

1. 根据接口定义里的数据结构、数据类型，自动生成 mock 规则。
2. 内置智能 mock 规则库，根据字段名、字段数据类型，智能优化自动生成的 mock 规则。
3. 可自动识别出图片、头像、用户名、手机号、网址、日期、时间、时间戳、邮箱、省份、城市、地址、IP 等字段，从而 Mock 出非常人性化的数据。
4. 支持自定义规则库，满足各种个性化需求。支持使用 正则表达式、通配符 来匹配字段名自定义 mock 规则。

# 自定义 Mock 规则

\* 名称

Pet

\* 分组

宠物相关

备注

JSON / XML 智能识别/快捷导入

预览

生成代码

根节点

object

Mock

中文名

说明

+

id

integer

Mock

宠物ID

说明

×

+

category

@ctitle

中文标题

分类

说明

×

+

name

@cword

中文词组

宠物名称

说明

×

+

photoUrls

@cparagraph

中文大段文本

照片URL

说明

×

+

items

@csentence

中文句子

中文名

说明

×

+

tags

@cname

中文姓名

标签

说明

×

+

items

@cfirst

中文姓

中文名

说明

+

status

@clast

中文名

销售状态

说明

×

+

createTime

@image

图片链接

创建时间

说明

×

+

location

string

Mock

位置

说明

×

+

Location

Mock

位置

说明

×

+

1. 支持 Mock.js 语法
2. 扩展身份证、国内手机号等常用规则

# 高级 Mock

项目概览

GET 查询宠物详情 × + ...

Mock 服务 ▾ ≡

文档 修改文档 运行 高级 Mock

期望 脚本

+ 新建期望

	启用 ①	名称	条件	创建人	操作
≡	<input checked="" type="checkbox"/>	异常状态宠物	Path 参数 petId 等于 1	apifox111	<a href="#">详情</a> <a href="#">快捷请求</a> <a href="#">...</a>
≡	<input checked="" type="checkbox"/>	在售宠物	Path 参数 petId 等于 2	apifox111	<a href="#">详情</a> <a href="#">快捷请求</a> <a href="#">...</a>
≡	<input checked="" type="checkbox"/>	记录不存在	Path 参数 petId 等于 3	apifox111	<a href="#">详情</a> <a href="#">快捷请求</a> <a href="#">...</a>
≡	<input type="checkbox"/>	ID格式不正确	Path 参数 petId 等于 DDD	apifox111	<a href="#">详情</a> <a href="#">快捷请求</a> <a href="#">...</a>

## 编辑

\* 期望名称

异常状态宠物

期望条件

	参数位置	参数名	比较	参数值	
≡	path ▾	petId	等于 ▾	1	×
	▾		等于 ▾		

返回数据

Body Header 更多设置

JSON Raw ① 支持 Mock.js、Nunjucks 语法

自动生成 预览 格式化

```
1 {
2   "id": 1,
3   "name": 123,
4   "status": "no",
5   "category": {
6     "id": 1999,
7     "name": "猫"
8   },
9   "photoUrls": [
10    "http://dummyimage.com/500x500"
11  ],
12  "tags": [
13  ]
14 }
```

取消

保存

# Mock 自定义脚本

项目概览GET 查询宠物详情+...

Mock 服务

文档修改文档运行高级 Mock

期望脚本

是否开启 ☐ 关闭

自定义脚本

格式化

```
1 // 获取自动 Mock 出来的数据
2 var responseJson = fox.mockResponse.json();
3
4 // 修改 responseJson 里的分页数据
5 // 将 page 设置为请求参数的 page
6 responseJson.page = fox.mockRequest.getParam('page');
7 // 将 total 设置 120
8 responseJson.total = 120;
9
10 // 将修改后的 json 写入 fox.mockResponse
11 fox.mockResponse.setBody(responseJson);
```

使用帮助

可通过自定义脚本获取用户请求参数, 修改返回内容。

示例代码

```
// 获取自动 Mock 出来的数据
var responseJson =
fox.mockResponse.json();

// 修改 responseJson 里的分页
数据
// 将 page 设置为请求参数的
page
responseJson.page =
fox.mockRequest.getParam('p
age');
// 将 total 设置 120
responseJson.total = 120;

// 将修改后的 json 写入
fox.mockResponse
fox.mockResponse.setBody(re
sponseJson);
```

请求对象: fox.mockRequest

```
fox.mockRequest.getParam(key:
```

保存

# 自动化测试

# 自动化测试

宠物商店

测试用例

测试套件

测试报告

接口管理

自动化测试

项目设置

在线分享

返回

普通用户下单流程测试

删除

编辑

分组

默认分组

创建时间

3 个月前

创建人

apifox111

最后修改时间

23 天前

优先级

P2

最后修改人

apifox111

测试步骤

测试数据

测试报告

已选 12 项

移除

步骤名称	绑定	操作
GET 查询宠物详情 (在售宠物)	未绑定	设置 复制 移除
GET 根据状态查找宠物列表 (出售中的宠物列表)	未绑定	设置 复制 移除
POST 上传宠物图片 (成功)	未绑定	设置 复制 移除
GET 登录 (成功)	未绑定	设置 复制 移除
POST 新建宠物信息 (成功)	已绑定	设置 复制 移除
PUT 修改宠物信息 (参数有误)	未绑定	设置 复制 移除
GET 根据状态查找宠物列表 (已售出宠物列表)	未绑定	设置 复制 移除
GET 查询用户信息 (查询用户信息)	未绑定	设置 复制 移除
GET 用户榜单 (用户榜单)	未绑定	设置 复制 移除
GET 查询宠物详情 (已售出宠物)	已绑定	设置 复制 移除
GET 查询用户信息 (普通用户)	已绑定	设置 复制 移除

\* 运行环境

Mock 服务

测试数据

停用

循环次数

1

线程数

1

间隔停顿

0

毫秒

保存变量变化值

使用全局 Cookie

保存 Cookie 到全局

运行

保存

持续集成

导出

从【接口用例】导入

Q

Y

全部 (24)

宠物 (12)

GET 查询宠物详情 (4) /pet/{petId}

已售出宠物

ID 格式不正确

在售宠物

记录不存在

删除宠物信息 (1) /pet/{petId}

新建宠物信息 (1) /pet

根据状态查找宠物列表 (2) /pet/findByStatus

根据 tag 查找宠物列表 (1) /pet/findByTags

上传宠物图片 (1) /pet/{petId}/uploadImage

修改宠物信息 (2) /pet

已选 0 个接口用例

导入模式

复制

绑定

取消

确认

# 测试数据

[< 返回](#)

普通用户下单流程测试

删除

编辑

分组 默认分组

创建时间 11 天前

创建人 apifox111

最后修改时间 12 分钟前

优先级 P2

最后修改人 apifox111

测试步骤

测试数据

测试报告

默认数据

测试环境

正式环境

Mock 服务









1. 每个数据集可包含多个变量，接口运行时 [使用变量](#) 的地方会读取对应的值（变量优先级：临时变量 > 测试数据变量 > 环境变量 > 全局变量）。
2. 可创建多个数据集，系统会遍历运行所有的数据集（每个数据集都会被运行一次）。
3. 查看 [《测试数据使用文档》](#)

添加数据集(行)

批量编辑

导入/导出

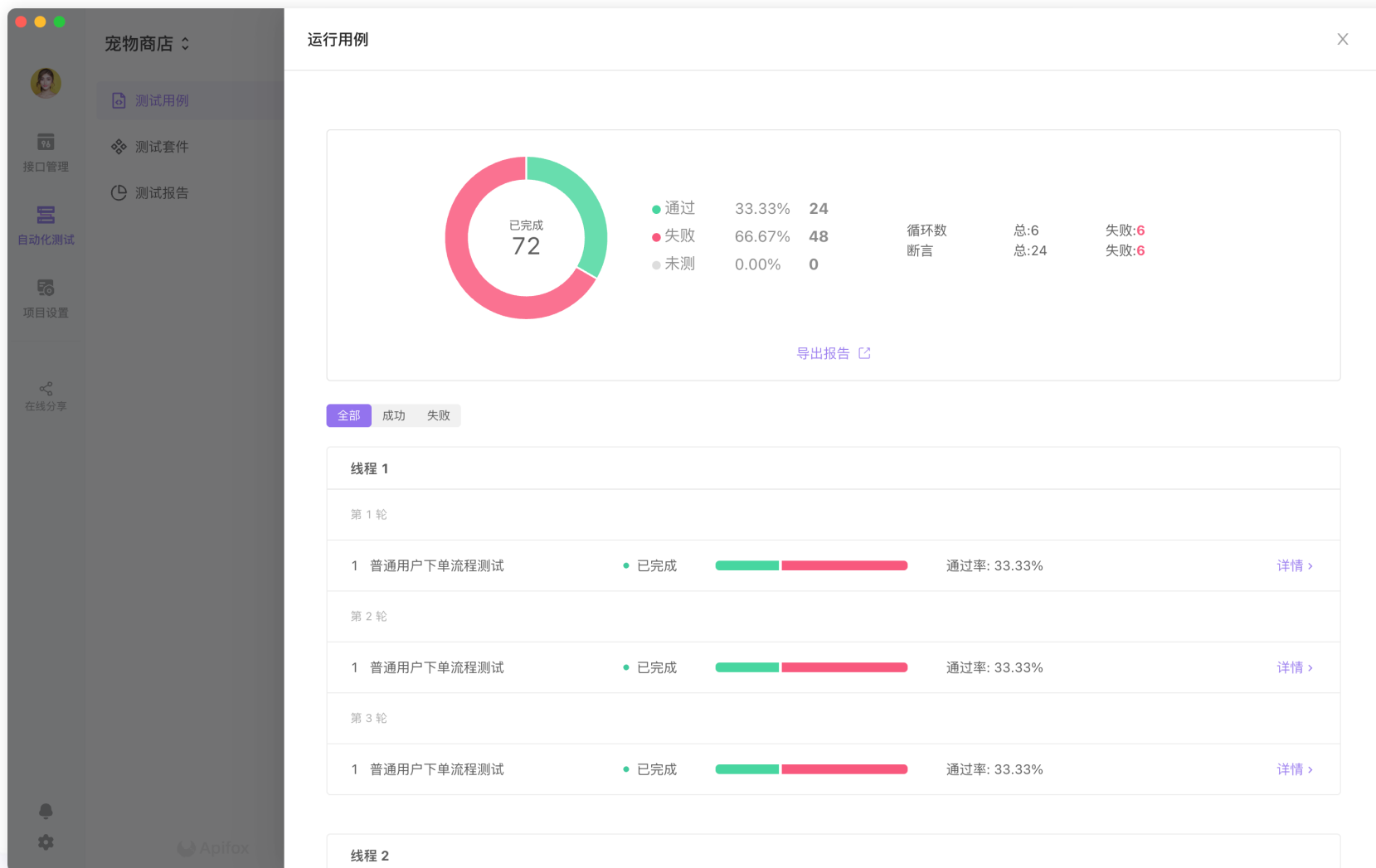


数据集名称	account	password	添加变量(列)
账户 1	test1@qq.com	123456	 
账户 2	test2@qq.com	123456	 
账户 3	test3@qq.com	123456	 
账户 4	test4@qq.com	123456	 

第 1-4 条/总共 4 条

&lt; 1 &gt; 20 条/页

# 自动化测试

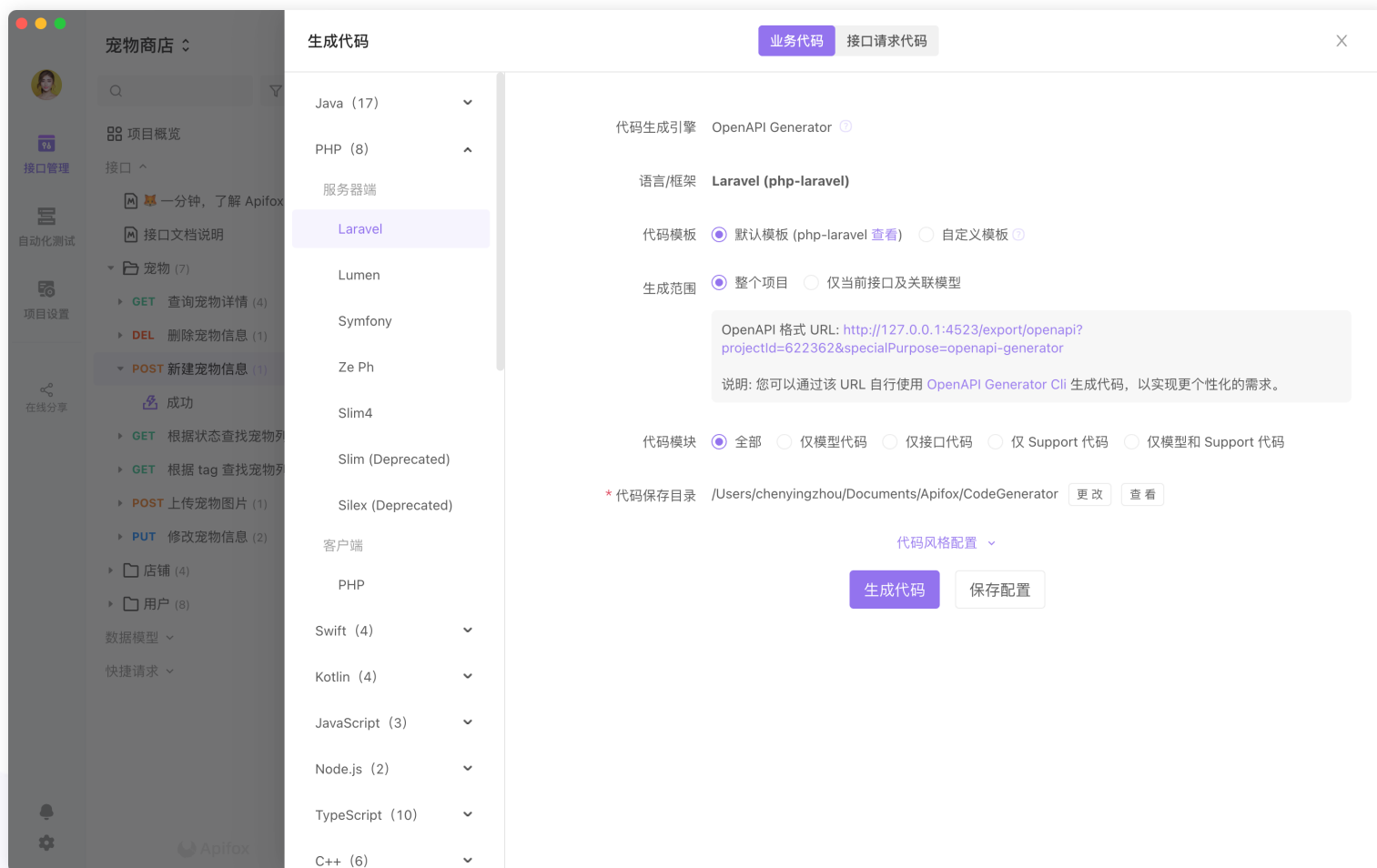


# 自动化测试



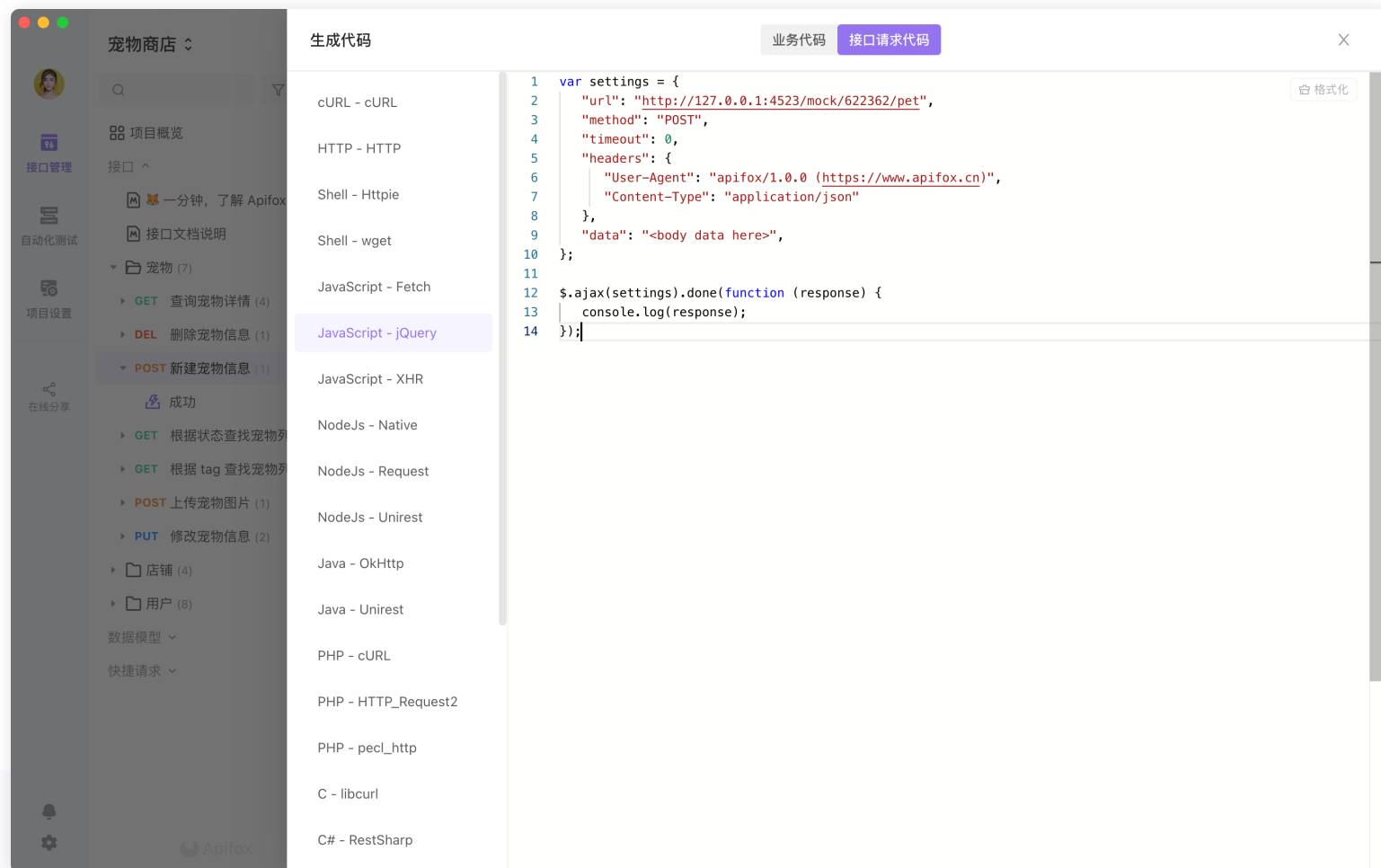
# 其他特性

# 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。

# 生成接口请求代码



宠物商店

项目概览

接口管理

自动化测试

项目设置

在线分享

一分钟，了解 Apifox

接口文档说明

宠物 (7)

- GET 查询宠物详情 (4)
- DEL 删除宠物信息 (1)
- POST 新建宠物信息 (1)

成功

- GET 根据状态查找宠物列
- GET 根据 tag 查找宠物列
- POST 上传宠物图片 (1)
- PUT 修改宠物信息 (2)

店铺 (4)

用户 (8)

数据模型

快速请求

生成代码

业务代码 接口请求代码

cURL - cURL

HTTP - HTTP

Shell - Httpie

Shell - wget

JavaScript - Fetch

JavaScript - jQuery

JavaScript - XHR

NodeJs - Native

NodeJs - Request

NodeJs - Unirest

Java - OkHttp

Java - Unirest

PHP - cURL

PHP - HTTP\_Request2

PHP - pecl\_http

C - libcurl

C# - RestSharp

```
1 var settings = {
2   "url": "http://127.0.0.1:4523/mock/622362/pet",
3   "method": "POST",
4   "timeout": 0,
5   "headers": {
6     "User-Agent": "apifox/1.0.0 (https://www.apifox.cn)",
7     "Content-Type": "application/json"
8   },
9   "data": "<body data here>",
10 };
11
12 $.ajax(settings).done(function (response) {
13   console.log(response);
14 });
```

格式化

# 自动生成代码

- 根据接口/模型定义，自动生成各种语言/框架的业务代码和接口请求代码。
- 支持 TypeScript、Java、Go、Swift、ObjectiveC、Kotlin、Dart、C++、C#、Rust 等 130 种语言及框架。
- 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。

# 支持 CI/CD

- 支持命令行方式运行接口测试 (Apifox CLI)。
- 支持集成 Jenkins 等持续集成工具。

```
$ apifox run examples/sample.apifox-cli.json -r cli,html
```

# 数据导入/导出

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

# 团队协作

- 接口数据云端同步，实时更新。
- 成熟的团队/项目权限管理，支持管理员、普通成员、只读成员等角色设置，满足各类企业的需求

# Thanks.

Apifox.cn

节省研发团队的每一分钟