

# APifox使用帮助

官方地址: <https://apifox.com>

官方帮助: <https://docs.apifox.com>

此帮助文档主要介绍如何简单介绍使用APifox调试接口，以及其常用功能。

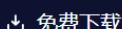
## 1 安装APifox

# API 设计、开发、测试一体化协作平台

Apifox = Postman + Swagger + Mock + JMeter

 API 文档  API 调试  API Mock  API 自动化测试

通过这里下载安装

 免费下载

使用 Web 版



他提供了网页版，要体验完整功能建议用客户端版本

安装过程非常简单，直接下一步、下一步就行，这里就不在截图了。

## 2 注册与登录

首次启动客户端，可以看到下图所示的效果

# 欢迎使用 Apifox

为什么需要登录?

手机号

邮箱

+86 ▾

手机号

验证码

获取短信验证码

点击“登录/注册”表示您同意 [服务协议](#) 和 [隐私协议](#)

登录/注册

密码登录

 微信登录

使用Apifox需要注册一个账号，选择一种你喜欢的方式注册与登录即可。

注册完成后，进入 Apifox 主界面。如果是首次登录，你将需要选择你的角色和工作模式。请根据你的工作职责选择最合适的选择。

# 欢迎进入 Apifox 的新世界 🎉

×

让我们通过下面的问题来构建您的工作流！

您的工作岗位是？

- 后端开发
- 全栈工程师
- 前端开发
- 产品/项目经理
- 测试
- 其他
- 技术 Leader

开始

如果你处于学习阶段，暂时没有明确的角色，可以选择“全栈工程师”和“API 设计优先”模式。

# 欢迎进入 Apifox 的新世界 🎉

×

让我们通过下面的问题来构建您的工作流！

您的工作岗位是？

- 后端开发
- 全栈工程师
- 前端开发
- 产品/项目经理
- 测试
- 其他
- 技术 Leader

您的团队开发模式是？

先写 API 文档，再写代码

先写代码，再输出 API 文档

开始

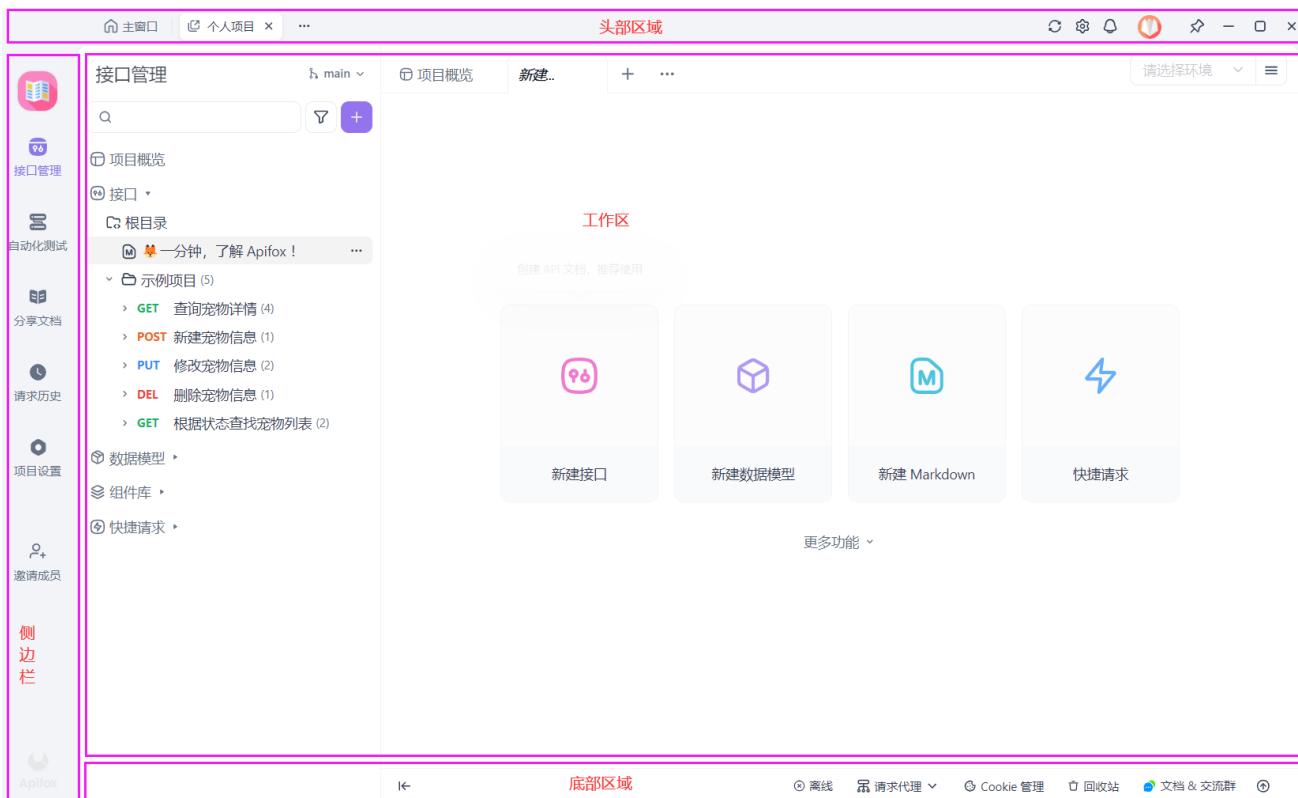
点击“开始”按钮，即可开始使用 Apifox。

## 3 页面布局

登录进来后，你会默认拥有一个个人团队，并且系统会默认在个人团队创建一个个人项目，如下图所示

The screenshot shows the Apifox application's main dashboard. At the top, there is a navigation bar with the Apifox logo, a 'Main Window' button (which is highlighted with a pink border), 'Personal Projects', and a more options menu. On the left side, there is a sidebar with sections for 'My Teams' (showing 'Personal Team' selected), 'API Hub' (with a note to discover more public projects), 'Favorites', and 'Recent Visits'. The main content area is titled 'Personal Team' and shows tabs for 'Team Projects', 'Team Resources', 'Activity', 'Members/Permissions', 'Order Management', and 'Team Settings'. A prominent feature is a large 'Personal Project' card, which includes a 'HTTP' icon and a 'Personal Project' title. This card is also highlighted with a pink border.

点击打开个人项目，简单介绍一下项目页面的布局



## 头部区域

头部区域包含以下功能模块：

- 主窗口
  - 进入个人主页，展示你加入的团队及其成员、项目、权限、订单、组织和 API Hub 等信息。
- 项目标签
  - 每个标签代表一个项目，点击即可切换；点击项目名称左侧按钮可在新窗口打开项目。
- 刷新
  - 重新加载当前项目数据，同时关闭所有打开的标签。
- 设置
  - 调整本地客户端的设置（如外观、网络代理等），这些设置仅影响当前客户端。
- 通知
  - 显示团队活动的提醒信息。
- 头像
  - 账号设置，可以设置基本账户信息、访问令牌（Access tokens）等。

## 侧边栏

侧边栏包含以下功能模块：

- 项目图标
  - 点击跳转到团队与项目。
- 接口管理
  - Apifox 的核心界面，你可以在这里 新建接口、发送接口请求、新建数据模型、编写 Markdown 等。

- 自动化测试
  - 如果你需要批量发送请求（类似运行 Postman 的 Collection）或编排接口之间的数据关系和逻辑，可以在自动化测试中新建测试场景。基于测试场景，你还可以查看测试报告、运行性能测试、管理测试数据、集成 CI/CD 等。
- 分享文档
  - 一旦创建接口，可以在该模块中分享给其他同事，或者发布文档站。发布文档站可以自定义文档域名、调整页面结构以及自定义 URL 等。
- 请求历史
  - 在请求历史中可以查看并重新发送所有已发送的请求。
- 项目设置
  - 所有与当前项目相关的设置，包括基本设置、功能设置、通知设置、项目资源和项目数据的导入/导出。
- 邀请成员
  - 邀请其他用户加入当前项目。

## 工作区

工作区主要包含目录树和标签页

### 目录树



## 项目概览

### 接口 ▾

#### 根目录

一分钟，了解 Apifox！

#### 示例项目 (5)

- > **GET** 查询宠物详情 (4)
- > **POST** 新建宠物信息 (1)
- > **PUT** 修改宠物信息 (2)
- > **DEL** 删除宠物信息 (1)
- > **GET** 根据状态查找宠物列表 (2)

#### 数据模型 ▾

#### 组件库 ▾

#### 快捷请求 ▾

目录树从上到下包括以下功能模块：

- 分支与版本
  - 默认显示主分支，你可以切换到其它分支，或者新建分支、管理分支。
- 搜索与过滤
  - 你可以搜索和过滤接口。注意，目前只能搜索接口名称，无法搜索目录名称或用例名称。
- 新建
  - 你可以创建各种类型的元素，如接口、快捷请求、数据模型、Markdown 等。
- 项目概览
  - 项目的整体视图，包括项目统计、接口用例覆盖情况等。
- 接口
  - 以目录结构组织的接口和接口用例，Markdown 和 WebSocket 等也可以包含在其中。
- 数据模型
  - 以目录结构组织的数据模型。
- 组件库
  - 可复用的数据模型，可设置默认响应模板以及响应组件。

- 快捷请求
  - 以目录结构组织的接口请求，类似于 Postman 的 Collection。

## 标签页

点击目录树中的任何元素会打开一个标签页



标签页包括以下类型：

- 接口
- 数据模型
- 响应组件
- Markdown
- 目录
- 快捷请求
- `WebSocket`
- 项目概览
- 新建...

单击打开标签页时，标题显示为斜体，此时点击目录树其它元素会覆盖当前标签，适合浏览场景。

修改标签内容后，标题变为常规字体，此时点击目录树其它元素会打开新标签，适合编辑场景。

若不希望标签被覆盖，可双击打开标签，使其直接显示为常规字体。

标签页的右侧有两个按钮：

- `+新建`: 可以创建各种类型的元素。
- `...更多`: 你可以关闭所有标签、关闭当前标签或关闭除当前标签之外的所有标签。

## 底部区域

底部区域包含以下功能模块：

- 折叠/展开
  - 可以折叠或展开左侧目录树。
- 在线状态
  - 显示当前是否在线。如果离线，团队同步将出现问题。
- Agent
  - `Apifox Web` 独有的功能，允许你选择代理发送请求。
- Cookie 管理
  - Cookie 管理器，可以存储当前的 Cookies。
- 回收站
  - 删除的接口、用例、数据模型、响应组件、测试场景、定时任务等会进入回收站，30 天后自动删除。
- 文档 & 交流群
  - 可打开帮助文档或加入交流群。

## 4 基本概念

### 文档模式/调试模式

Apifox 的 API 调试有两种模式，可以接口标签页左下角切换，分别是：文档模式 和 调试模式。

The screenshot shows the Apifox interface with the 'Document Mode' tab selected. On the left, the sidebar shows a tree structure of API endpoints under the '示例项目 (5)' section. The first endpoint, 'GET /pet/{petId}' (labeled '查询宠物详情'), is highlighted with a red box. On the right, the main panel displays the details for this endpoint, including its method (GET), path, status (已发布), and creation date (2025年6月17日). Below this, the 'Mock' section and '请求参数' (Request Parameters) section are visible. At the bottom of the main panel, there is a navigation bar with a back arrow, a '文档模式' button (highlighted with a red box), and a '调试模式' button.

这两种模式提供了类似的功能，但界面不同，以适应不同团队的工作流程。

- **文档模式**是 Apifox 推荐的模式，适合采用 API 设计优先的团队。在这个模式中，团队首先定义 API，随后根据 API 文档进行开发和测试。



- **调试模式**则特别适合那些没有事先定义 API 文档的团队。这样的团队通常集中在后端开发，先完成代码，然后再生成 API 文档进行测试工作。



如果需要调用其他人开发的 API 而没有文档，建议也使用调试模式。

## 接口

Apifox 是一个以 API (接口) 为核心的产品，这意味着所有操作都从定义 API 开始。

The screenshot shows the Apifox interface. On the left, there's a sidebar with '接口管理' (API Management) and a search bar. The main area shows a 'GET' request for 'http://127.0.0.1:4532/m1/6597672-6303364-default/pet/{petId}' with a status of '成功 (200)'. It includes sections for 'Query 参数' (Query Parameters), 'Path 参数' (Path Parameters), and a 'Body' tab showing a JSON response:

```
5 "name": "次面区",
6 "photoUrls": [
7     "http://dummyimage.com/200x200"
8 ],
9 "category": {
10     "id": 8318988253471762,
11     "name": "Cat"
12 },
```

- 在 Apifox 的主界面中，接口是基本元素，它们以目录的形式进行分组。对于每个接口，你可以进行修改、预览、发送请求，或者将请求保存为接口用例。
- 这种结构与 Postman 大不相同，Apifox 更像是一种基于 OAS (即 *OpenAPI Specification*, 一种用于描述和定义 RESTful API 的标准化格式) 的扩展——你可以直接调试和保存请求。
- 在 Postman 中，基本元素是“请求”，而这些请求与 API 定义 (即 *API specification*, 是定义和描述 API 行为的关键文档或标准) 本身是分离的。这意味着当 API 定义发生变化时，所有的请求和脚本都需要重新编写。
- 在 Apifox 中，所有的接口用例 (对应 Postman 的请求) 都是基于 API 定义 (*API specification*) 的。当 API 定义发生变化时，接口用例会同步变化，所有基于此的测试场景和 CI/CD 也可以自动或手动更新，非常适合开发团队在维护和更新 API 时使用。

## 快捷请求

在 Apifox 中，你也可以创建快捷请求。这些快捷请求不需要基于 API 定义进行设计，其方式与 Postman 请求相同。你还可以将成功的请求保存为接口文档。

The screenshot shows the Apifox interface with a '快捷请求' (Quick Request) tab selected. It has fields for 'URL' (输入 http 或 https 起始的完整 URL) and '发送' (Send). Below are tabs for 'Params', 'Body', 'Headers', etc., and a 'Query 参数' (Query Parameters) section.

## 测试场景

当你需要批量发送请求时 (类似运行 Postman Collection)，你可以使用自动化测试来编排测试场景。

The screenshot shows the Apifox interface with a sidebar on the left containing icons for '接口管理' (API Management), '自动化测试' (Automated Testing), '分享文档' (Share Document), '请求历史' (Request History), '项目设置' (Project Settings), and '邀请成员' (Invite Members). The main area is titled '自动化测试' (Automated Testing) and shows a '测试步骤' (Test Steps) section for a '测试宠物新建/修改/删除流程' (Test Pet Create/Edit/Delete Flow). The steps listed are:

- POST 新建宠物信息 (新建在售宠物)
- GET 查询宠物详情 (查询刚刚建的宠物)
- PUT 修改宠物信息 (修改宠物状态为 sold)
- GET 查询宠物详情 (查询宠物状态是否修改成功)
- DELETE 删除宠物信息 (删除该宠物)
- GET 查询宠物详情 (检查删除是否成功)

On the right side, there is a '功能测试' (Functional Testing) panel with sections for '运行环境' (Run Environment), '测试数据' (Test Data), '循环次数' (Loop Count), '运行于' (Run On), and '通知' (Notification). Buttons for '运行' (Run) and '保存' (Save) are at the bottom.

一个测试场景包括一系列测试步骤。这些测试步骤可以从接口或接口用例中导入到测试场景，并可以在接口文档发生变化时自动或手动更新相应的参数。

测试场景还支持流程控制条件如 If、for、forEach 等。你可以在测试步骤间传递数据、动态生成请求参数。

基于测试场景，你还可以查看测试报告、运行性能测试、管理测试数据、集成 CI/CD 等。

## 环境

Apifox 中的环境与 Postman 中的环境类似，包含许多变量。切换环境时，可以使用相同环境变量的不同值。

The screenshot shows the '环境管理' (Environment Management) dialog. On the left, a sidebar lists environments: 全局 (Global), 开发环境 (Development Environment), 测试环境 (Testing Environment), 正式环境 (Production Environment), 体验环境 (Experience Environment), 本地 Mock (Local Mock) (selected), 云端 Mock (Cloud Mock), and 自托管 Mock (Self-hosted Mock). On the right, the '本地 Mock' tab is selected, showing a '服务 (前置URL)' (Service (Frontend URL)) section with a service named '默认服务' (Default Service) and URL 'http://127.0.0.1:4523/m1/6597672-6303364-default'. Below it is a '环境变量' (Environment Variables) table with columns: 变量名 (Variable Name), 类型 (Type), 远程值 (Remote Value), 本地值 (Local Value), and 说明 (Description). A button for '添加变量' (Add Variable) is also present.

然而，Apifox 的环境还包括另一个重要概念：服务。每个服务对应一个前置 URL。由于 OpenAPI/Swagger 中定义的接口都以 / 开头，它们可以被发送到不同的服务。在 Apifox 中，你无需在请求开头写 {{Base\_url}} 来设置服务为变量。只需切换到对应的环境，所有请求都会自动发送到该环境中的相应服务。

# 项目

接口、快捷请求、数据模型、环境、测试场景等的集合构成了一个项目。在 Apifox 中，项目是协作的基本单位。

一个项目对应一个 API 定义 (*API specification*)。你还可以将 OpenAPI/Swagger 导入项目，或将项目导出为 OpenAPI/Swagger。

与 Postman 相比，Apifox 项目大致相当于 Postman 的 Collection，而 Apifox 团队则类似于 Postman 的 Workspace。

## 5 接口操作

Apifox 为 API 设计者提供了直观的界面来新建接口。下面来演示一个简单的分页查询示例

### 新建接口

新建接口有很多入口，选择一个你喜欢的方式都可以，如下图圈出来的地方都可以新建接口



顺便提一下，为了方便分模块或归类管理接口，你可以创建目录来管理你的接口，避免很混乱，通过下面的菜单可以创建目录

主窗口 个人项目 X ...

## 接口管理

main

+

...

项目概览

接口

根目录

一分钟，了解 Apifox！

示例项目 (5)

数据模型

组件库

快捷请求

+ 新建

添加子目录

添加 Markdown

添加 WebSocket 接口

新建 Socket.IO 接口

添加 Socket 服务

导入

导出

生成业务代码

点击完新建接口后，会新开一个标签页来用于编辑新的接口的相关信息，当然在没有点击保存之前不会将接口保存到目录树的。

The screenshot shows a user interface for creating a new API endpoint. At the top, there are tabs for 'GET 新建接口' (New Endpoint), '+ ...', and '本地 Mock'. On the right, there are buttons for '本地 Mock' and a three-dot menu. Below the tabs, there's a search bar with 'GET' selected and a placeholder '接口路径, "/"起始'. A purple '保存' (Save) button is on the right.

The main area is titled '未命名接口' (Unnamed Endpoint). It has four sections: '可见性 \*' (Visibility) set to '共享' (Shared), '状态 \*' (Status) set to '开发中' (In Development), '责任人' (Owner) with a dropdown, and '标签' (Tags) with a search input '查找或回车创建标签'. Below these is a section for '服务 (前置URL)' (Service (Frontend URL)) with options '继承父级' (Inherit from parent) and '跟随父...' (Follow parent...).

Under the '说明' (Description) section, there's a rich text editor with a placeholder '支持 Markdown 格式' (Supports Markdown format). At the bottom of this section are buttons for '+ 自定义字段' (Custom Fields) and '+ OAS 扩展' (OAS Extensions).

接下来我们来定义一下接口的定义请求信息，主要包括：

1. 请求方式：常用的请求方式 (GET、POST、PUT、DELETE)
2. 请求地址：在接口路径栏中输入请求地址，请求地址/开头，如 /user/query-all
3. 接口名称：给接口命名，这样我们才知道此接口是做什么业务的
4. 接口说明：可以给接口添加详细的描述信息
5. 请求参数：因为查询一般是GET请求，所以我们设置的是查询参数

GET 查询用户列表

+ ...

本地 Mock

≡

接口 修改接口 运行 Mock

⌚ ⚡ ⚡ ⚡ ⚡

1 GET /user/query-all 2

保存

运行

删除

## 查询用户列表 3

可见性 \*

共享

状态 \*

• 开发中

责任人

标签

服务 (前置URL) ①

继承父级

跟随父...

说明

这是一个示例接口，用于表达一个简单的查询接口 4

+ 自定义字段 + OAS 扩展

### 请求参数 5

Params 3

Body Headers Cookies Auth 前置操作 后置操作 设置

Query 参数

参数名	类型	示例值	说明	更多
pageIndex	= integer *	1	查询数据页号	更多
pageSize	= integer *	10	查询数据条数	更多
name	= string *	zs	用户名称	更多 S

添加参数

输入完成后记得保存，保存成功后可以在，目录树种看到你的接口了

然后再来定义响应结构，滚动至 返回响应 部分，在“成功(200)”响应下，点击“+”添加字段。

#### 返回响应

##### 成功(200)

+ 添加

HTTP 状态码: 200 名称: 成功 内容格式: JSON application/json

数据结构

预览 生成代码 JSON Schema 添加子节点

根节点

object

Mock

中文名 说明

添加子节点

没有字段 添加

+ 添加示例

+ 添加描述

+ Headers

+ OAS 扩展

比如下面是我添加的响应结构示例

## 返回响应

成功(200)

+ 添加

HTTP 状态码: 200 名称: 成功 内容格式: JSON ▾ application/json

数据结构

预览 生成代码 JSON Schema 通过 JSON 等生成

根节点		object	Mock	中文名	说明	编辑
code	integer *	10000	中文名	状态码	+	-
message	string *	操作成功	中文名	提示信息	+	-
data	object *	Mock	中文名	分页数据	...	-
pageIndex	integer *	10	中文名	页码	+	-
pageSize	integer *	Mock	中文名	数据条数	+	-
total	integer *	10	中文名	数据数	+	-
rows	array *	Mock	中文名	当前页数据	+	-
ITEMS	object	Mock	中文名	数据项	...	-
id	string *	Mock	中文名	ID 编号	+	-
name	string *	Mock	中文名	名称	+	-
age	string *	Mock	中文名	年龄	+	-

数据类型自己调整

字段描述, 可以用mock指定动态值, 也可以是固定值



+ 添加示例

+ 添加描述

+ Headers

+ OAS 扩展

修改完记得随手保存一下。

有了响应结构可以使用它帮你生成响应示例, 点击添加响应示例

id	string *	(\$string.uuid)	中文名	ID 编号	编辑
name	string *	(\$person.fullName)	中文名	名称	编辑
age	string *	(\$number.int(min=18	中文名	年龄	编辑

+ 添加示例

+ 添加描述

+ Headers

+ OAS 扩展

然后再弹出界面生成响应示例

## 添加示例

X

示例名称

成功示例

1

示例值

2

自动生成

格式化

```
1  {
2      "code": 10000,
3      "message": "操作成功",
4      "data": [
5          {
6              "pageIndex": 5,
7              "pageSize": 10,
8              "total": 96,
9              "rows": [
10                 {
11                     "id": "94b68591-cf4e-40e3-92bf-5c2ee5db6ea3",
12                     "name": "巨国琴",
13                     "age": "40"
14                 },
15                 {
16                     "id": "17ff7eb8-6d84-4acf-ad35-11f63c76573d",
17                     "name": "母建军",
18                     "age": "47"
19             }
20         ]
21     ]
22 }
```

+ 描述 + OAS 字段名称 + OAS 扩展

取消

3 确定

点击确定后可以看到响应示例，如下图所示

示例

```
{
    "code": 10000,
    "message": "操作成功",
    "data": [
        {
            "pageIndex": 5,
            "pageSize": 10,
            "total": 96,
            "rows": [
                {
                    "id": "94b68591-cf4e-40e3-92bf-5c2ee5db6ea3",
                    "name": "巨国琴",
                    "age": "40"
                },
                {
                    "id": "17ff7eb8-6d84-4acf-ad35-11f63c76573d",
                    "name": "母建军",
                    "age": "47"
                }
            ]
        }
    ]
}
```

有了响应示例可以方便，前后端对接的时候看具体响应数据格式，保存后切换到接口预览视图，可以看到接口的完整信息

The screenshot shows a user interface for managing API endpoints. At the top, there's a header with a 'GET' button labeled '查询用户列表', a '+' button, and a '本地 Mock' dropdown. Below the header, there are tabs for '接口' (Interface), '修改接口' (Edit Interface), '运行' (Run), and 'Mock'. On the right side of the header are icons for refresh, search, and other operations.

The main content area has a title '查询用户列表' (Query User List) and a sub-section '请求参数' (Request Parameters). Under 'Query 参数', there are three parameters listed:

- `pageIndex` integer (必填) - Description: 查询数据页号, Example value: 1
- `pageSize` integer (必填) - Description: 查询数据条数, Example value: 10
- `name` string (可选) - Description: 用户名称, Example value: zs

Below this, under '返回响应' (Return Response), there's a section for '成功(200)'. It shows the HTTP status code (200), content type (application/json), and the data structure of the response. The response structure is as follows:

数据结构	示例
<code>code</code> integer 状态码 <code>message</code> string 提示信息 ▼ <code>data</code> object 分页数据 <ul style="list-style-type: none"><li><code>pageIndex</code> integer 页码</li><li><code>pageSize</code> integer 数据条数</li><li><code>total</code> integer 数据数</li><li>► <code>rows</code> array [object { }] 当前页数据</li></ul>	{ "code": 1000, "message": "操作成功", "data": { "pageIndex": 5, "pageSize": 10, "total": 96, "rows": [ { "id": "94b68591-cf4e-40e3-92bf-5c2ee5db6ea3", "name": "巨国琴", "age": "40" }, { "id": "17ff7eb8-6d84-4acf-ad35-11f63c76573d", "name": "母建军", "age": "47" } ] } }

# 发送接口请求

当接口创建完成后，即可发起请求。

1. 在“**文档模式**”下，切换到“**运行**”标签页。
2. 在右上角选择环境为“**本地 Mock**”。
3. 点击“**发送**”按钮以发送请求。
4. 你可以在**返回响应区域**查看请求的结果。

The screenshot shows the interface of a local API testing tool. At the top, there's a header with 'GET' and '查询用户列表'. Below it, a toolbar has a 'Run' button (highlighted with a red circle) and an 'Environment' dropdown set to 'Local Mock' (also highlighted with a red circle). The main area shows a GET request URL: 'http://127.0.0.1:4523/m1/6597672-6303364-default/user/query-all'. A 'Send' button is highlighted with a red circle. Below the URL, tabs for 'Params' (selected), 'Body', 'Headers', 'Cookies', 'Auth', and 'Settings' are visible. Under 'Params', there are three query parameters: 'pageIndex' (integer, value 1), 'pageSize' (integer, value 10), and 'name' (string, value 'zs'). A 'Send' button is also present here. At the bottom, a large section titled 'Body' displays the JSON response. The response is: { "code": 10000, "message": "操作成功", "data": { "pageIndex": 1, "pageSize": 10, "total": 97, "rows": [ { "id": "156f7cf2-9ebd-4f7d-bf4c-3b13b46c404e", "name": "碧宝轩" } ] } }. This JSON is highlighted with a pink rectangle and a red circle labeled '4' is placed on the 'rows' key. To the right of the response, there's a status bar showing '200 135 ms 248 B' and a validation result box stating '校验响应结果' and '返回数据结构与接口定义一致'.

切换到“**实际请求**”以查看实际发送的请求。

Body Cookie Header 8 控制台

实际请求 •



请求 URL:

**GET** http://127.0.0.1:4523/m1/6597672-6303364-default/user/query-all?  
pageIndex=1&pageSize=10&name=zs

Header:

名称	值
User-Agent	Apifox/1.0.0 (https://apifox.com)
Accept	*/*
Host	127.0.0.1:4523
Accept-Encoding	gzip, deflate, br
Connection	keep-alive

...  
更多

更多内容[点击访问官网](#), 学习了解。