# API管理服务器搭建

# API 简介

网络应用程序，分为前端和后端两个部分。当前的发展趋势，就是前端设备层出不穷（手机、平板、桌面电脑、其他专用设备......）。

因此，必须有一种统一的机制，方便不同的前端设备与后端进行通信。而API就是完成和前端和后端的交互的地方。这导致API构架的流行，甚至出现"API First"的设计思想。API协议就是前端和后端完成交互时所遵循的规则。目前，RESTful API是比较成熟的一套互联网应用程序的API设计理论。

## API接口 组成

* 接口描述

这个接口是用来干嘛的，以及相关的规则。

* 接口地址：

以网址的形式展现，你通过发送请求给这个网址来对接口进行交互操作

* 请求方法：

常用的有GET/POST/DELETE/PUT

* 请求参数：

请求该接口时，需要提供的参数，参数属性包括名称、类型、是否必填、描述等

* 返回参数：

接口正常响应后，返回的内容

* 错误代码：

接口请求失败后，返回的错误代码

## RestfulApi的组成

### 1.2.1 通信协议

API与用户的通信协议，总是使用HTTPs协议。

### 1.2.2 域名

应该尽量将API部署在专用域名之下。

|  |
| --- |
| https://api.bjsxt.com |

如果确定API很简单，不会有进一步扩展，可以考虑放在主域名下:

|  |
| --- |
| https://bjsxt.com/api/ |

### 1.2.3版本

应该将API的版本号放入URL。

|  |
| --- |
| https://api.bjsxt.com/v1/ |

另一种做法是，将版本号放在HTTP头信息中，但不如放入URL方便和直观。Github采用这种做法。

### 1.2.4 路径(Endpoint)

路径又称"终点"（endpoint），表示API的具体网址。

在RESTful架构中，每个网址代表一种资源（resource），所以网址中不能有动词，只能有名词，而且所用的名词往往与数据库的表格名对应。一般来说，数据库中的表都是同种记录的"集合"（collection），所以API中的名词也应该使用复数。

举例来说，有一个API提供学院（academy）的信息，还包括各种学生和老师的信息，则它的路径应该设计成下面这样。

* https://api.bjsxt.com/v1/academies
* https://api.bjsxt.com/v1/students
* https://api.bjsxt.com/v1/teachers

### 1.2.5 HTTP动词

对于资源的具体操作类型，由HTTP动词表示。

常用的HTTP动词有下面五个（括号里是对应的SQL命令）。

* GET（SELECT）：从服务器取出资源（一项或多项）。
* POST（CREATE）：在服务器新建一个资源。
* PUT（UPDATE）：在服务器更新资源（客户端提供改变后的完整资源）。
* PATCH（UPDATE）：在服务器更新资源（客户端提供改变的属性）。
* DELETE（DELETE）：从服务器删除资源。

还有两个不常用的HTTP动词。

* HEAD：获取资源的元数据。
* OPTIONS：获取信息，关于资源的哪些属性是客户端可以改变的。

下面是一些例子。

* GET /academies：列出所有学院
* POST /academies：新建一个学院
* GET /academies/ID：获取某个指定学院的信息
* PUT /academies/ID：更新某个指定学院的信息（提供该学院的全部信息）
* PATCH /academies/ID：更新某个指定学院的信息（提供该学院的部分信息）
* DELETE /academies/ID：删除某个学院
* GET /academies/ID/students：列出某个指定学院下的所有学员
* DELETE /academies/ID/students/ID：删除某个指定学院的指定学员

### 1.2.6 过滤信息(Filtering)

如果记录数量很多，服务器不可能都将它们返回给用户。API应该提供参数，过滤返回结果。

下面是一些常见的参数:

|  |
| --- |
| ?limit=10：指定返回记录的数量  ?offset=10：指定返回记录的开始位置。  ?page=2&num=100：指定第几页，以及每页的记录数。  ?sortby=name&order=asc：指定返回结果按照哪个属性排序，以及排序顺序。  ?academy\_type\_id=1：指定筛选条件 |

参数的设计允许存在冗余，即允许API路径和URL参数偶尔有重复。比如，GET /academies/ID/students与 GET /students?academy\_id=ID 的含义是相同的。

### 1.2.7 状态码（Status Codes）

服务器向用户返回的状态码和提示信息，常见的有以下一些（方括号中是该状态码对应的HTTP动词）。

* 200 OK - [GET]：服务器成功返回用户请求的数据，该操作是幂等的（Idempotent）。
* 201 CREATED - [POST/PUT/PATCH]：用户新建或修改数据成功。
* 202 Accepted - [\*]：表示一个请求已经进入后台排队（异步任务）
* 204 NO CONTENT - [DELETE]：用户删除数据成功。
* 400 INVALID REQUEST - [POST/PUT/PATCH]：用户发出的请求有错误，服务器没有进行新建或修改数据的操作，该操作是幂等的。
* 401 Unauthorized - [\*]：表示用户没有权限（令牌、用户名、密码错误）。
* 403 Forbidden - [\*] 表示用户得到授权（与401错误相对），但是访问是被禁止的。
* 404 NOT FOUND - [\*]：用户发出的请求针对的是不存在的记录，服务器没有进行操作，该操作是幂等的。
* 406 Not Acceptable - [GET]：用户请求的格式不可得（比如用户请求JSON格式，但是只有XML格式）。
* 410 Gone -[GET]：用户请求的资源被永久删除，且不会再得到的。
* 422 Unprocesable entity - [POST/PUT/PATCH] 当创建一个对象时，发生一个验证错误。
* 500 INTERNAL SERVER ERROR - [\*]：服务器发生错误，用户将无法判断发出的请求是否成功。

### 1.2.8 错误处理（Error handling）

如果状态码是4xx/5xx，就应该向用户返回出错信息。一般来说，返回的信息中将error作为键名，出错信息作为键值即可。

|  |
| --- |
| {  "error": "error message"  } |

### 1.2.9 返回结果

针对不同操作，服务器向用户返回的结果应该符合以下规范。

* GET /collection：返回资源对象的列表（数组）
* GET /collection/resource：返回单个资源对象
* POST /collection：返回新生成的资源对象
* PUT /collection/resource：返回完整的资源对象
* PATCH /collection/resource：返回完整的资源对象
* DELETE /collection/resource：返回一个空文档

### 1.2.10 其他

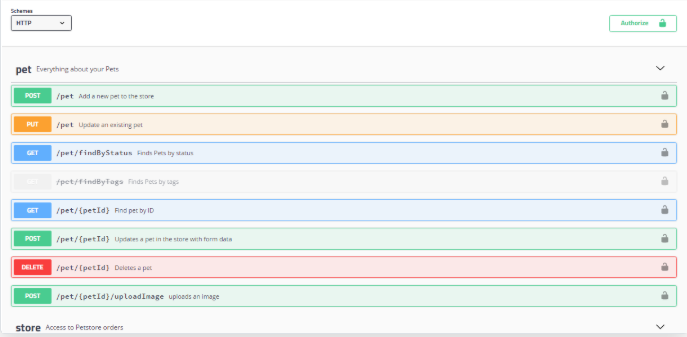
（1）API的身份认证应该使用OAuth 2.0框架。

（2）服务器返回的数据格式，应该尽量使用JSON，避免使用XML。

# Swagger2简介

Swagger 是一套围绕 OpenAPI 规范构建的开源工具，可以帮助您设计，构建，记录和使用 REST API。

Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。Swagger 的目标是对 REST API 定义一个标准的和语言无关的接口，可让人和计算机无需访问源码、文档或网络流量监测就可以发现和理解服务的能力。当通过 Swagger 进行正确定义，用户可以理解远程服务并使用最少实现逻辑与远程服务进行交互。



我们在项目中集成Swagger，编写项目的swagger注解后，项目就会生成一份swagger接口，但是我们将项目关闭后，前端就无法访问该文档了，因此，我们还需要给前端搭建一套API的接口展示服务，减少前端与后端的沟通成本。

# YAPI 简介

YApi 是高效、易用、功能强大的 api 管理平台，旨在为开发、产品、测试人员提供更优雅的接口管理服务。可以帮助开发者轻松创建、发布、维护 API，YApi 还为用户提供了优秀的交互体验，开发人员只需利用平台提供的接口数据写入工具以及简单的点击操作就可以实现接口的管理。

**特性**

* 基于 Json5 和 Mockjs 定义接口返回数据的结构和文档，效率提升多倍
* 扁平化权限设计，即保证了大型企业级项目的管理，又保证了易用性
* 不仅有类似 postman 的接口调试，还有强大的测试集功能
* 免费开源，内网部署，信息再也不怕泄露了！
* 支持 postman, har, swagger 数据导入

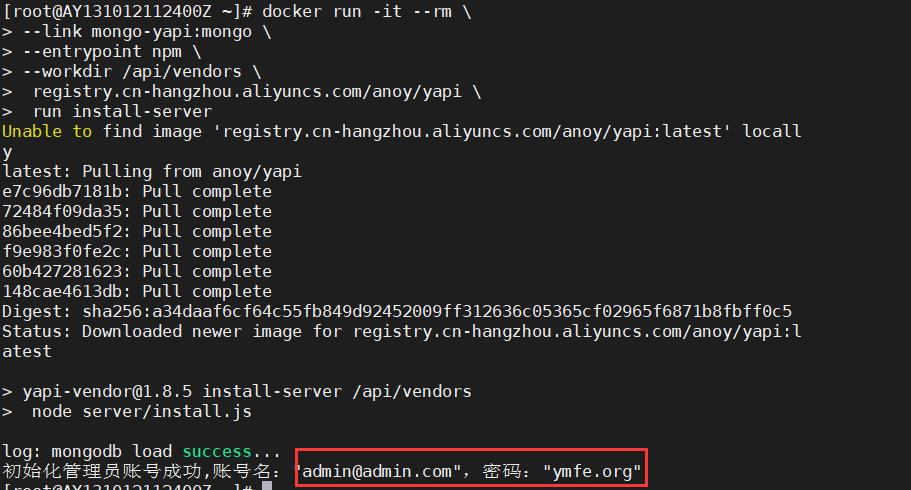
## 3.1 YApi的私有云部署

启动YApi-mongo：

|  |
| --- |
| docker run -d --name mongo-yapi mongo |

初始化YApi：

|  |
| --- |
| docker run -it --rm \  --link mongo-yapi:mongo \  --entrypoint npm \  --workdir /api/vendors \  registry.cn-hangzhou.aliyuncs.com/anoy/yapi \  run install-server |



启动YApi：

|  |
| --- |
| docker run -d \  --name yapi \  --link mongo-yapi:mongo \  --workdir /api/vendors \  -p 3000:3000 \  registry.cn-hangzhou.aliyuncs.com/anoy/yapi \  server/app.js |

## 3.2 YApi的访问

浏览器打开:

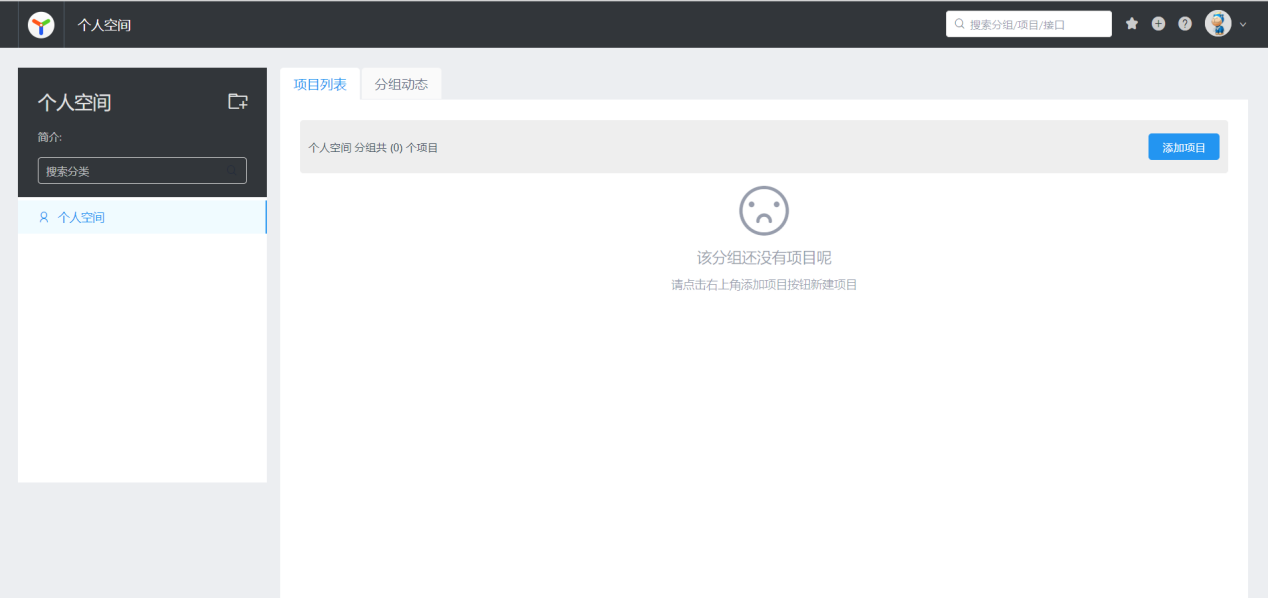
|  |
| --- |
| http://你的公网ip:3000/ |



可以选择直接使用默认的账号登录，或者是注册一个新的账号。

在这里我选择使用默认的账号登录：

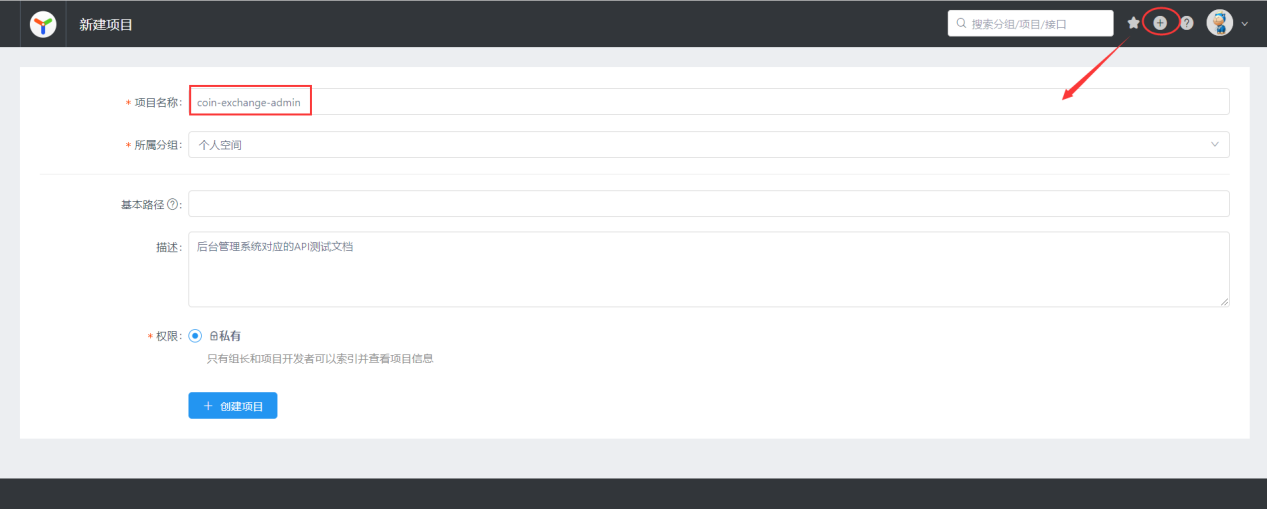




## 3.3 CoinExchange接口文档的导入

### 3.3.1 CoinExchangeAdmin

新建项目：



导入数据：



选择这个:



### 3.3.2 CoinExchangeUser

新建项目



导入数据:

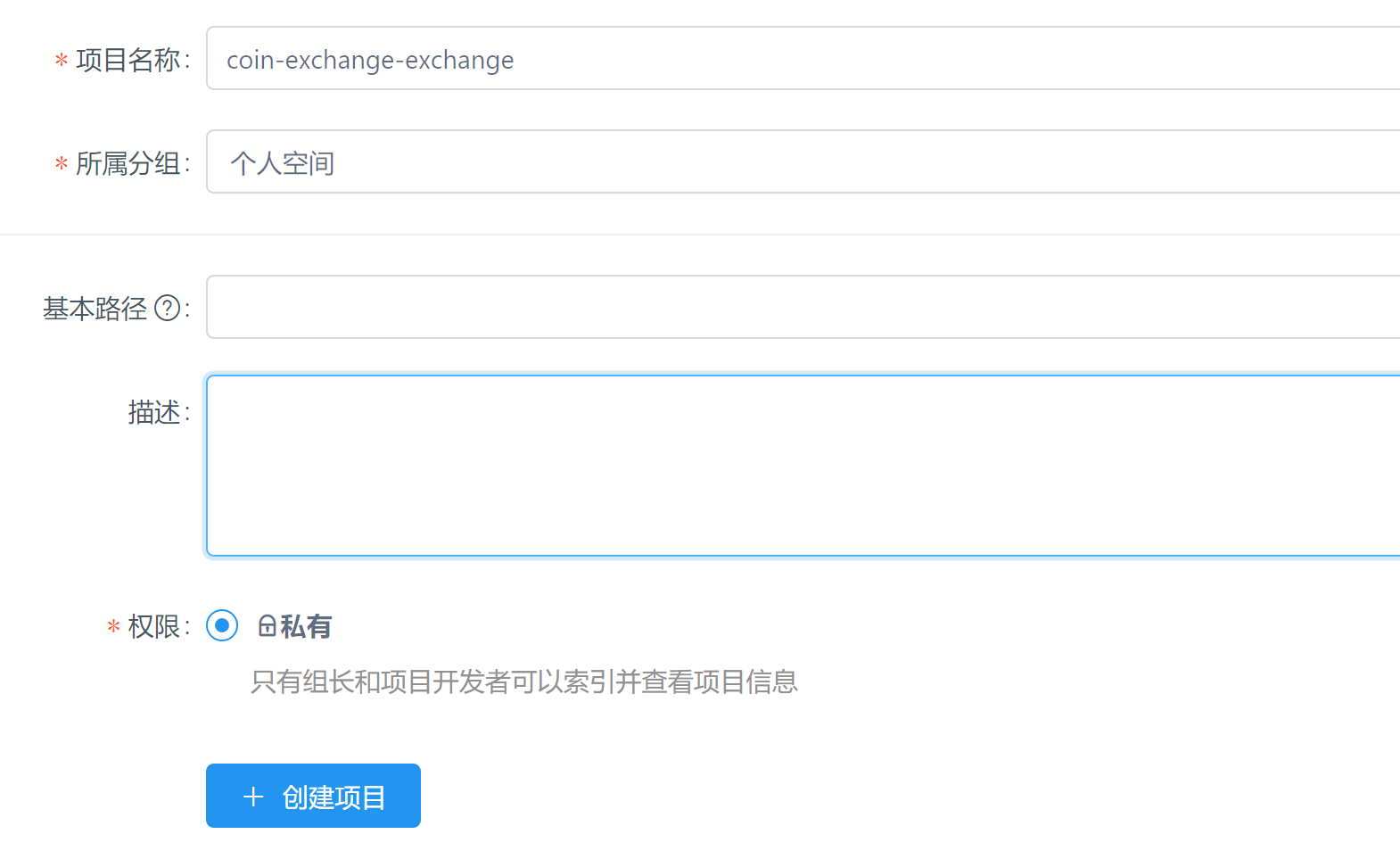


选择文件:



### 3.3.3 CoinExchangeExchange

新建项目:



导入数据:



选择文件:

