

# 生成 Swagger 在线文档

## 本节核心内容

- 如何给 API 添加 Swagger 文档功能
- 如何编写 API 文档

本小节源码下载路径: [demo17](#)

([https://github.com/lexkong/apiserver\\_demos/tree/master/demos/swagger](https://github.com/lexkong/apiserver_demos/tree/master/demos/swagger))

可先下载源码到本地, 结合源码理解后续内容, 边学边练。

本小节的代码是基于 [demo16](#)

([https://github.com/lexkong/apiserver\\_demos/tree/master/demos/swagger](https://github.com/lexkong/apiserver_demos/tree/master/demos/swagger)) 来开发的。

## 背景

开发 API 服务, API 文档必不可少, 很多人选择手写 API 文档, 手写 API 文档有很多问题, 比如工作量大、手写容易出错、更新麻烦、格式不固定、维护困难等。所以在实际的开发中, 建议自动生成 API 文档。

本小册所用的 API 服务器 RESTful 框架采用的是 gin, gin 在 GitHub 上有很多 middleware 可用, 其中就有可以自动生成 Swagger 文档的 gin-swagger middleware。

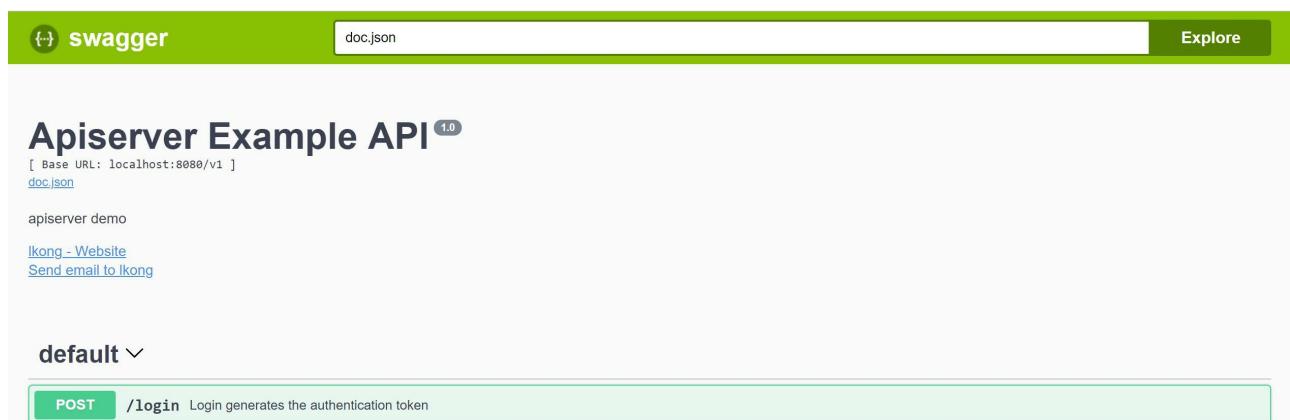
# Swagger 简介

Swagger 是一个强大的 API 文档构建工具，可以自动为 RESTful API 生成 Swagger 格式的文档，可以在浏览器中查看 API 文档，也可以通过调用接口来返回 API 文档（JSON 格式）。Swagger 通常会展示如下信息：

1. HTTP 方法（GET、POST、PUT、DELETE 等）
2. URL 路径
3. HTTP 消息体（消息体中的参数名和类型）
4. 参数位置
5. 参数是否必选
6. 返回的参数（参数名和类型）
7. 请求和返回的媒体类型

Swagger 还有一个强大的功能：可以通过 API 文档描述的参数来构建请求，测试 API。

浏览器访问截图：



swagger

doc.json

Explore

Apiserver Example API <sup>1.0</sup>

[ Base URL: localhost:8080/v1 ]

[doc.json](#)

apiserver demo

[Ikong - Website](#)

[Send email to lkong](#)

default ▾

POST /login Login generates the authentication token

JSON 返回截图：

```
[api@centos user]$ curl http://127.0.0.1:8080/swagger/doc.json
{
  "swagger": "2.0",
  "info": {
    "description": "apiserver demo",
    "title": "Apiserver Example API",
    "contact": {
      "name": "lkong",
      "url": "http://www.swagger.io/support",
      "email": "466701708@qq.com"
    },
    "license": {},
    "version": "1.0"
  },
  "host": "localhost:8080",
  "basePath": "/v1",
  "paths": {
    "/login": {
      "post": {
        "produces": [
          "application/json"
        ]
      }
    }
  }
}
```

## Swagger 配置步骤

我们用 [gin-swagger \(https://github.com/swaggo/gin-swagger\)](https://github.com/swaggo/gin-swagger) gin middleware 来生成 Swagger API 文档。步骤如下：

### 1. 安装 swag 命令

```
$ mkdir -p $GOPATH/src/github.com/swaggo
$ cd $GOPATH/src/github.com/swaggo
$ git clone https://github.com/swaggo/swag
$ cd swag/cmd/swag/
$ go install -v
```

因为该包引用 [golang.org](https://golang.org) 中的包，而网络环境原因，一般很难连上 [golang.org](https://golang.org)，所以这里不采用 go get 方式安装。

swag 的依赖包已经包含在第 4 节的 vendor 包下。

### 2. 进入 apiserver 根目录执行 swag init

```
$ cd $GOPATH/src/apiserver
$ swag init
```

### 3. 下载 gin-swagger

```
$ cd $GOPATH/src/github.com/swaggo
$ git clone https://github.com/swaggo/gin-swagger
```

### 4. 在 router/router.go 中添加 swagger 路由 (详见 [demo17/router/router.go](#) [https://github.com/lexkong/apiserver\\_demos/blob/master/router/router.go](https://github.com/lexkong/apiserver_demos/blob/master/router/router.go))



### 5. 编写 API 注释, Swagger 中需要将相应的注释或注解编写到方法上, 再利用生成器自动生成说明文件

这里用创建用户 API 来举例, 其它 API 请参考

[demo17/handler/user](#)

[https://github.com/lexkong/apiserver\\_demos/tree/master/controllers](https://github.com/lexkong/apiserver_demos/tree/master/controllers) 下的 API 文件。

```
package user

import (
    ...
)

// @Summary Add new user to the database
// @Description Add a new user
// @Tags user
// @Accept json
// @Produce json
// @Param user body user.CreateRequest true
// "Create a new user"
// @Success 200 {object} user.CreateResponse "
// {"code":0,"message":"OK","data": {
// "username":"kong"}}
// @Router /user [post]
func Create(c *gin.Context) {
    ...
}
```

6. 执行 `swag init`, 在 `apiserver` 根目录下生成 `docs` 目录

```
$ swag init
```

## 文档语法说明

- `Summary`: 简单阐述 API 的功能
- `Description`: API 详细描述
- `Tags`: API 所属分类
- `Accept`: API 接收参数的格式
- `Produce`: 输出的数据格式, 这里是 JSON 格式

- Param: 参数, 分为 6 个字段, 其中第 6 个字段是可选的, 各字段含义为:
  1. 参数名称
  2. 参数在 HTTP 请求中的位置 (body、path、query)
  3. 参数类型 (string、int、bool 等)
  4. 是否必须 (true、false)
  5. 参数描述
  6. 选项, 这里用的是 `default()` 用来指定默认值
- Success: 成功返回数据格式, 分为 4 个字段
  1. HTTP 返回 Code
  2. 返回数据类型
  3. 返回数据模型
  4. 说明
- 路由格式, 分为 2 个字段:
  1. API 路径
  2. HTTP 方法

API 文档编写规则请参考 [See Declarative Comments Format](#)  
([https://swaggo.github.io/swaggo.io/declarative\\_comments.html](https://swaggo.github.io/swaggo.io/declarative_comments.html))

API 文档有更新时, 要重新执行 `swag init` 并重新编译 `apiserver`。

## 编译并运行

1. 下载 `apiserver_demos` 源码包 (如前面已经下载过, 请忽略此步骤)

```
$ git clone  
https://github.com/lexkong/apiserver_demos
```

2. 将 `apiserver_demos/demo17` 复制为  
`$GOPATH/src/apiserver`

```
$ cp -a apiserver_demos/demo17/  
$GOPATH/src/apiserver
```

3. 在 `apiserver` 目录下编译源码

```
$ cd $GOPATH/src/apiserver  
$ make
```

执行 `./apiserver` 启动 `apiserver` 后，在浏览器中打开：  
`http://localhost:8080/swagger/index.html` 访问  
Swagger 2.0 API 文档。

API 总览：

Apiserver Example API (1.0)

[ Base URL: localhost:8080/v1 ]

[doc.json](#)

apiserver demo

[Ikong - Website](#)  
[Send email to Ikong](#)

**default** ▾

**POST** /login Login generates the authentication token

**sd** ▾

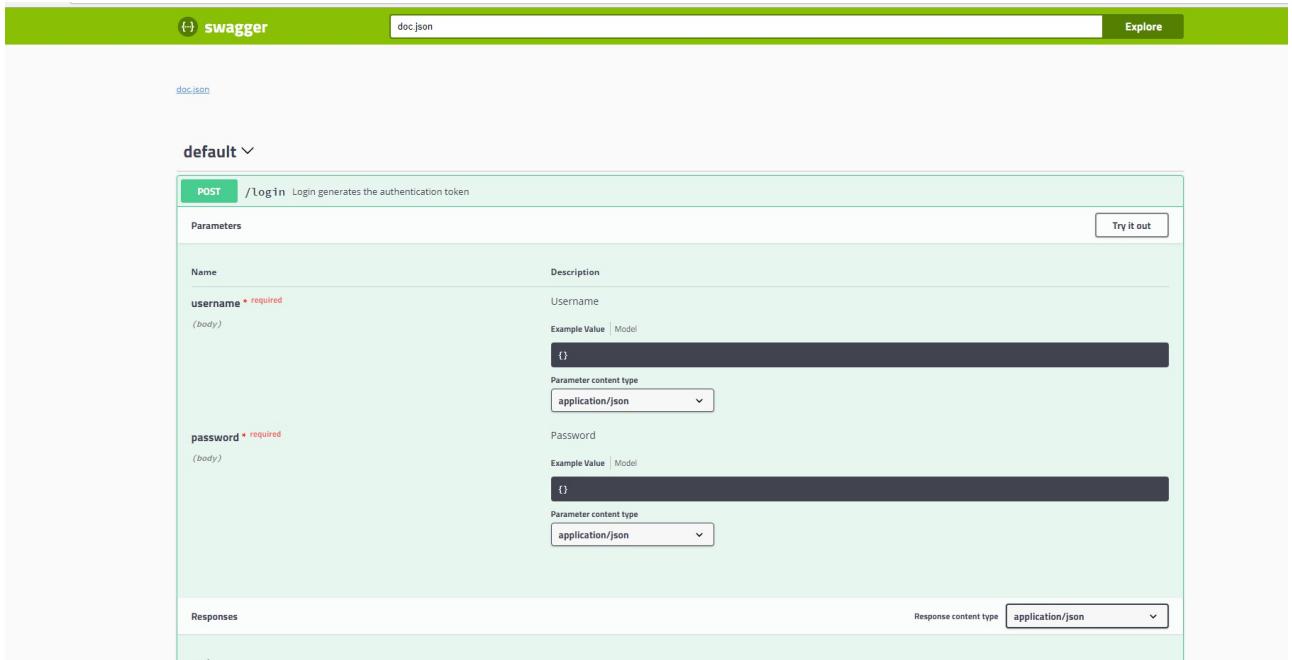
**GET** /sd/cpu Checks the cpu usage

**GET** /sd/disk Checks the disk usage

**GET** /sd/health Shows OK as the ping-pong result

**GET** /sd/ram Checks the ram usage

点击 `/login`, 查看 `login` API 详情：



The screenshot shows the Swagger UI interface for a POST /login endpoint. The endpoint description is "Login generates the authentication token". It has two required parameters: "username" and "password", both of type application/json. The "username" example value is {} and the "password" example value is also {}.

## 小结

本小节介绍了如何生成 Swagger 文档，并演示了具体的效果。本小节也是动手操作的最后一个小节，至此恭喜你成功构建了一个企业级的 API 服务器，[demo17](#) ([https://github.com/lexkong/apiserver\\_demos/tree/master/c](https://github.com/lexkong/apiserver_demos/tree/master/c)) 即为此 API 服务器的最终源码。