



《手写OS操作系统》小班二期招生，全程直播授课，大牛带你掌握硬核技术！

点此查看

[慕课网首页](#) [免费课](#) [实战课](#) [体系课](#) [慕课教程](#) [专栏](#) [手记](#) [企业服务](#)[🔍](#) [🛒](#) [🔔](#) [我的](#)[从所有教程的词条中查询...](#)[首页](#) > [慕课教程](#) > [Go工程师体系课全新版](#) > [4. JSON、ProtoBuf 渲染](#)

全部开发者教程

[4. 库存的微服务设计](#)[5. 商品分类的各个函数](#)[6. 轮播图的各个handler](#)[7. 品牌相关的接口](#)[8. 品牌分类接口](#)[第12周 商品微服务的gin层和oss图片服务](#)[1. 商品列表接口](#)[2. 新建商品](#)[3. 删除，更新商品](#)[4. 商品分类接口](#)[5. 轮播图接口](#)[6. 品牌分类](#)[7. oss快速入门](#)[8. 阿里云oss开发入门](#)[9. 服务端签名直传](#)[10. 内网穿透](#)[11. 项目文档和资源](#)

bobby · 更新于 2022-11-08

[← 上一节](#) [3. 获取参数](#) [5. 表单验证](#) [下一节 →](#)

1. 输出json和protobuf

新建user.proto文件

<> 代码块

```
1 syntax = "proto3";
2 option go_package = ".;proto";
3
4 message Teacher {
5     string name = 1;
6     repeated string course = 2;
7 }
```

<> 代码块

```
1 package main
2
3 import (
4     "github.com/gin-gonic/gin"
5     "net/http"
6     "start/gin_t/proto"
7 )
8
9 func main() {
10     r := gin.Default()
11     // gin.H is a shortcut for map[string]interface{}
12     r.GET("/someJSON", func(c *gin.Context) {
13         c.JSON(http.StatusOK, gin.H{"message": "hey", "status": http.StatusOK})
14     })
15     r.GET("/moreJSON", func(c *gin.Context) {
16         // You also can use a struct
17         var msg struct {
18             Name    string `json:"user"`
19             Message string
20             Number   int
21         }
22         msg.Name = "Lena"
23         msg.Message = "hey"
24         msg.Number = 123
25         // Note that msg.Name becomes "user" in the JSON
26         // Will output : {"user": "Lena", "Message": "hey", "Number": 123}
27         c.JSON(http.StatusOK, msg)
28     })
29
30     r.GET("/someProtoBuf", func(c *gin.Context) {
31         courses := []string{"python", "django", "go"}
```

[📝 意见反馈](#)[💖 收藏教程](#)[🔖 标记书签](#)

ta/proto





```
34         Name: "bobby",
35         Course: courses,
36     }
37     // Note that data becomes binary data in the response
38     // Will output protoexample.Test protobuf serialized data
39     c.ProtoBuf(http.StatusOK, data)
40 })
41 // Listen and serve on 0.0.0.0:8080
42 r.Run(":8083")
43 }
```

2. PureJSON

通常情况下，JSON会将特殊的HTML字符替换为对应的unicode字符，比如 < 替换为 \u003c，如果想原样输出html，则使用PureJSON

<> 代码块

```
1 func main() {
2     r := gin.Default()
3
4     // Serves unicode entities
5     r.GET("/json", func(c *gin.Context) {
6         c.JSON(200, gin.H{
7             "html": "<b>Hello, world!</b>",
8         })
9     })
10
11    // Serves literal characters
12    r.GET("/purejson", func(c *gin.Context) {
13        c.PureJSON(200, gin.H{
14            "html": "<b>Hello, world!</b>",
15        })
16    })
17
18    // listen and serve on 0.0.0.0:8080
19    r.Run(":8080")
20 }
```



3. 获取参数 ◀ 上一节 下一节 ▶ 5. 表单验证

我要提出意见反馈

