

Go搭建REST服务

链家网 丁靖

自我介绍

丁靖

- 8年PHP, PECL开发
- Swoole开发组成员
- 半年Go开发
- 链家网存储/图片服务负责人
- 微信/微博: samt42



Go搭建REST服务

- Go编程之美
- REST API 设计原则
- REST 服务开发框架
- PHP To Go

TIOBE

20

95



Go

1.270%

+1.19%

TIOBE Index for Go

Source: www.tiobe.com



Go语言

- 强类型语言
- 非面向对象语言
- 并发语言
- 系统级语言

强类型

强类型

```
var a int = 10
var b int64 = 10
if a == b {
    fmt.Println("Equal")
} else {
    fmt.Println("Not Equal")
}
```

隐式类型转换

```
var a int = 10
```

```
var b int64 = 10
```

```
c := a10 + bb
```


非面向对象

没有类的概念

```
type Language struct {  
    Name string  
}  
func (l Language) Best() {  
    fmt.Println(l.Name + "是世界上最好的语言")  
}  
func main() {  
    v := Language{}  
    v.Name = "PHP"  
    v.Best()  
}
```

PHP是世界上最好的语言

没有继承

```
type AnotherLanguage struct {  
    Language  
}  
func (l AnotherLanguage) Best() {  
    fmt.Println(l.Name + "是世界上最最好的语言")  
}  
func main() {  
    v := AnotherLanguage{}  
    v.Name = "PHP"  
    v.Best()  
}
```

PHP是世界上最最好的语言

并发

异步

- 我正在撸代码
- 收到一个Bug
- 请A帮忙修复一下, 修完告诉我
- 我继续撸代码

异步

```
c := make(chan string) //channel
fix := func(man string) {
    fmt.Println(man + "修复Bug")
    fmt.Println(man + "撸代码")
    c <- man + "修复完成"
}
fmt.Println("撸代码")
fmt.Println("撸代码")
fmt.Println("收到一个Bug")
go fix("A")
for {
    select {
    case msg := <-c:
        fmt.Println(msg)
    default:
        fmt.Println("撸代码")
    }
}
```

并行

- 一边撸代码
- 一边听歌
- 持续一小时

并行

```
go func() {  
    for {  
        fmt.Println("撸代码")  
    }  
}()  
go func() {  
    for {  
        fmt.Println("听歌")  
    }  
}()  
select {  
    case <-time.After(1 * time.Hour):  
        fmt.Println("休息一下")  
}
```

网络

HttpServer

```
func echo(w http.ResponseWriter, r *http.Request) {  
    io.WriteString(w, "Hello world")  
}  
func main() {  
    http.HandleFunc("/", echo)  
    err := http.ListenAndServe(":9999", nil)  
    if err != nil {  
        log.Fatal("ListenAndServe: ", err)  
    }  
}
```


扩展 HttpServer

```
type MyServer struct {  
    *http.Server  
}  
  
func (srv *MyServer) ListenAndServe() error {  
    addr := srv.Addr  
    if addr == "" {  
        addr = ":http"  
    }  
    ln, err := net.Listen("tcp", addr)  
    if err != nil {  
        // added to write log  
        log.Println(os.Getpid(), srv.Addr)  
        return err  
    }  
    return srv.Serve(tcpKeepAliveListener{ln.(*net.TCPListener)})  
}
```

优点

- 比 PHP, Python 更接近系统
- 比 C/C++ 更容易上手
- 编译速度
- 跨平台/无依赖
- 为并发而生
- 丰富的系统库

REST API 设计原则

资源

- 为所有“资源”定义唯一URL标识

如:

<http://example.com/product/1> (商品1)

<http://example.com/picture/1.jpg> (图片1)

- “资源”关系用链接定义

```
<product ref="http://example.com/product/1">
```

```
    <image>http://example.com/picture/1.jpg</image>
```

```
</product>
```

表现

- HTTP方法
GET 获取资源 / POST 新建资源
PUT 更新或新增资源/ DELETE 删除资源
- 多重表示
如: 请求1:Accept: text/xml
 响应1:Content-Type:text/xml
 请求2:Accept: text/html
 响应2:Content-Type:text/html

经典 REST API

- AWS S3
- Github API

REST 服务开发框架



<https://github.com/go-ozzo/ozzo-routing>

组件

- ozzo-log
- ozzo-config
- ozzo-di
- ozzo-validation
- ozzo-routing

路由

```
router := routing.New()
api := router.Group("/api")

api.Get("/users", func(c *routing.Context) error {
    return c.Write("user list")
})

api.Put(`/users/<id:\d+>`, func(c *routing.Context) error {
    return c.Write("update user " + c.Param("id"))
})

http.Handle("/", router)
http.ListenAndServe(":8080", nil)
```

还要做什么？

- 异常处理
- 优雅退出/重启
- 请求超时
- 配置热更新
- ...

异常处理

修复 http.HandlerFunc 的 panic

```
func MyHandler(h http.Handler) http.Handler {  
    return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {  
        var err error  
        defer func() {  
            r := recover()  
            if r != nil {  
                http.Error(w, "error", http.StatusInternalServerError)  
            }  
        }()  
        h.ServeHTTP(w, r)  
    })  
}
```

优雅退出/重启

接管 http.Server.ConnState

```
#net/http/server.go
type Server struct {
.
.
// ConnState specifies an optional callback function that is
// called when a client connection changes state.
// 当连接状态改变时的回调
ConnState func(net.Conn, ConnState)
.
.
}
```

PHP To Go

PHP To Go

- 思维转变
 - 强类型
 - 异步/并行
- 并发带来的资源锁问题
 - 共享资源需要锁
 - 避免死锁

PHP To Go

- 常驻内存
 - 合理利用资源
 - 资源热更新
- 网络编程
 - 机遇和挑战

Q/A