Go自动生成SDK实践

Go语言北京用户组

27 February 2016

蒙卓 (mengzhuo)

项目代码

此代码仅为演示，没有版权，随意使用

https://github.com/mengzhuo/auto-gen-sdk-talks-at-bjgug

我们(要)推出Golang SDK啦!

API样式

Host Instance

Description	Interface
Start	StartInstance (./api/start_instance.html)
Stop	StopInstance (./api/stop_instance.html)
List all hosts	ListHost (./api/list_host.html)

还有100多个.....

如何编写SDK?

(复制粘贴，写API函数) X 100+次

呵呵

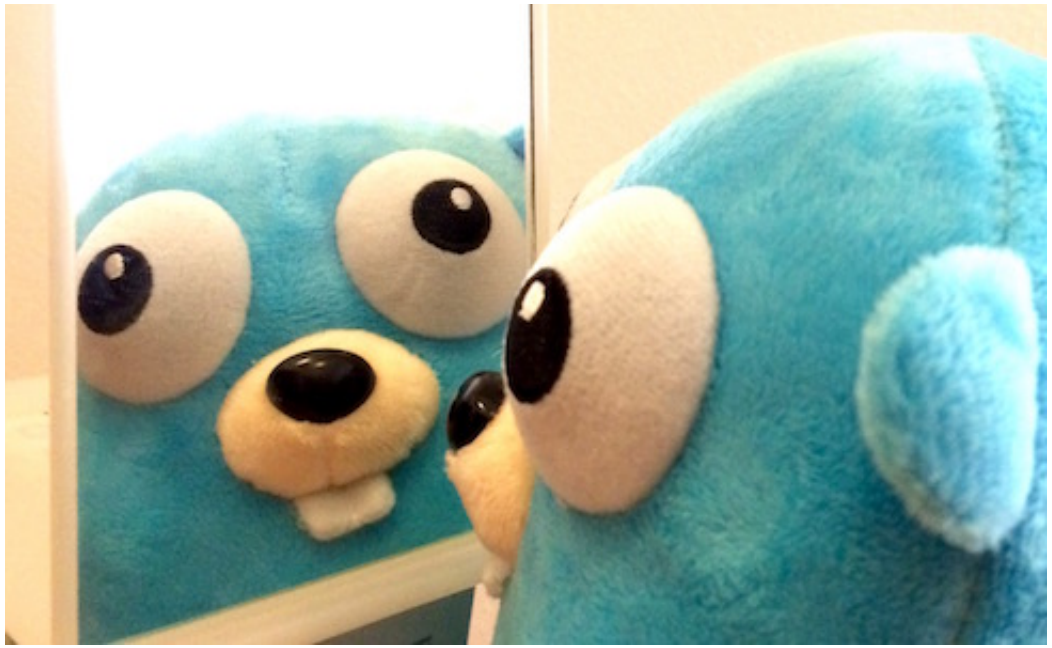
更进一步!

Reflect

定义：

Reflection in computing is the ability of a program to examine its own structure

Reflection（反射）就是程序有检查自身结构的能力



如何利用Reflect制作SDK

简介

```
val := reflect.ValueOf(obj) //获取obj的值
t := reflect.TypeOf(obj) //获取obj的类型
```

如何利用Reflect制作SDK

例子

```
package main

import (
    "fmt"
    "reflect"
)

type Cat struct {
    Color string
}

func main() {
    cat := &Cat{"red"}

    fmt.Println("值", reflect.ValueOf(cat))
    fmt.Println("类型", reflect.TypeOf(cat))
}
```

Run

如何利用Reflect制作SDK

构建Request类

```
type StartRequest struct {
    Region string `xc:"required`
    HostId string `xc:"required`
}
```

如何利用Reflect制作SDK

构建Request类

```
type StartRequest struct {
    Region string `xc:"required`
    HostId string `xc:"required`
}
```

- 由 ` `封闭起来的叫做Struct Tag
- 在reflect里可以用 Field().Tag.Get(<指定项>)获得

如何利用Reflect制作SDK

构建Response类

```
type StartRequest struct {
    Region string `xc:"required`
    HostId string `xc:"required`
}


type StartResponse struct {
    Action string
    HostId string
}
```

如何利用Reflect制作SDK

```
func Do(request Interface{}) {

    v := reflect.ValueOf(request)
    typ := reflect.TypeOf(request)

    params := make(map[string]string, 0)

    for i := 0; i < typ.NumField(); i++ {

        name := typ.Field(i).Name
        tag := typ.Field(i).Tag.Get("xc")
        field := v.Field(i)

        switch field.Kind() {
        case reflect.Int, reflect.Int8, reflect.Int16, reflect.Int32, reflect.Int64:
            num := field.Int()
            if num == 0 && tag == "optional" {
                continue
            }
            params[name] = strconv.FormatInt(num, 10)
        case reflect.String:
            str := field.String()
            params[name] = str
        }
    }
```

```
    typ_name_list := strings.Split(typ.String(), ".")
    typ_name := typ_name_list[len(typ_name_list)-1]
    err := u.Get(params, rsp)
    return rsp, err
}
```

如何利用Reflect制作SDK

总结

reflect可以获得类型和值

reflect tag可以进一步细化操作

如何利用Reflect制作SDK

缺点

- 运行速度慢
- 还是要复制粘贴一段时间
- 不太好debug： （

更进一步!

如何借助Go generate 生成SDK?

1. 编写生成代码
2. $go generate

Go generate 介绍

文件格式

```
//go:generate <cmd>
```

1. 搜索GOPATH下所有//go:generate开头的代码文件
2. 执行所有<cmd>

官方简介(https://blog.golang.org/generate)

咱不会yacc……

一些窍门

```
go run gen.go
```

- 巧用// +build ignore
- text/template
- go run 很快!

生成器讲解

gen.go
Elem对象

```
type Elem struct {
    Name     string
    Type     string
    Doc      string
    Required bool
}
```

生成器讲解

构建Request类
通过HTML解析，获取Elem

生成器讲解

构建Request类

```
type Elem struct {
    Name     string
    Type     string
    Doc      string
    Required bool
}
```

```
+--------+--------+------+----------+
|  Name  |  Type  | Doc  | Required |
+--------+--------+------+----------+
| Region | String | blah | Yes      |
| HostID | String | blah | Yes      |
+--------+--------+------+----------+
```

生成器讲解

使用Request类

```
func (r *{{.Name}}) GenURL() (v *url.Values) {
    setAuth(v)
    {{range $elem := .Parameters}}
    {{if eq $elem.Type "Integer" }}
        v.Set("{{ $elem.Name }}", intToString(r.{{$elem.Name}}))
    {{else}}
        v.Set("{{ $elem.Name }}", r.{{$elem.Name}})
    {{end}}
    {{end}}
    v.Set("Action", "{{.Name}}")
    return v
}
```

生成器讲解

生成的Request类的代码

```
/* 《我只是说明》
+--------+--------+------+----------+
|  Name  |  Type  | Doc  | Required |
+--------+--------+------+----------+
| Region | String | blah | Yes      |
| HostID | String | blah | Yes      |
+--------+--------+------+----------+
*/

func (r *StartInstance) GenURL() (v *url.Values) {
    setAuth(v)
    v.Set("Region", r.Region)
    v.Set("HostId", r.HostId)
    v.Set("Action", "StartInstance")
    return v
}
```

生成器讲解

Demo

生成器讲解

优点

- 容易debug
- 可以重复
- 不会复制粘贴错

Let the machine do the work -- Rob Pike

Happy Hacking Q&A

Thank you

蒙卓 (mengzhuo)
mengzhuo1203@gmail.com (mailto:mengzhuo1203@gmail.com)
@mengzhuo (http://twitter.com/mengzhuo)