链滴

golang 数据处理（七）

作者：GumKey

原文链接：https://ld246.com/article/1566446121334

来源网站：链滴

<font color="#006666">一、map、struct、json的互相转换</fnt>

1、struct转json

```
package main

import (
    "encoding/json"
    "fmt"
)

type Demo struct {
    Akey string
    Bkey int
}

func main() {
    // todo struct转json
    demo := Demo{Akey:"a", Bkey:1}
    jsons, errs := json.Marshal(demo) //转换成JSON返回的是byte[]
    if errs != nil {
        fmt.Println(errs.Error())
    }
    fmt.Println(string(jsons)) //byte[]转换成string 输出
}
```

2、json转struct

```
package main

import (
    "fmt"
    "encoding/json"
)

type Demo struct {
    Akey string
    Bkey int
}

func main() {
    // todo json转struct
    jsons :=`{"Akey":"a","Bkey":1}`
    var demo_s Demo
    errs :=json.Unmarshal([]byte(jsons), &demo_s)
    if errs != nil {
        fmt.Println(errs.Error())
    }
    fmt.Println(demo_s)

}
```

3、map转json

```
package main

import (
    "fmt"
    "encoding/json"
)

func main() {
    // todo map转json
    demoMap := map[string]interface{}{"Akey":"a", "Bkey":1}
    jsons, errs := json.Marshal(demoMap) //转换成JSON返回的是byte[]
    if errs != nil {
        fmt.Println(errs.Error())
    }
    fmt.Println(string(jsons)) //byte[]转换成string 输出
}
```

4、 json转map

```
package main

import (
    "fmt"
    "encoding/json"
)

func main() {
    // todo json转map
    var demoMap map[string]interface{}
    jsons :=`{"Akey":"a","Bkey":1}`

    errs :=json.Unmarshal([]byte(jsons), &demoMap)
    if errs != nil {
        fmt.Println(errs.Error())
    }
    fmt.Println(demoMap)

}
```

5、map转struct（或：map转json，json转struct)

```
package main

import (
    "fmt"
    "github.com/goinggo/mapstructure"
)

type Demo struct {
    Akey string
    Bkey int
```

```
}

func main() {
    // todo map转struct（map转json，json转struct)
    demoMap := map[string]interface{}{"Akey":"a", "Bkey":1}
    var demo Demo
    err := mapstructure.Decode(demoMap, &demo)
    if err != nil {
        fmt.Println(err)
    }
    fmt.Println(demo)
}
```

6、struct转map（或：struct转json， json转map）

```
package main

import (
    "fmt"
    "reflect"
)

type Demo struct {
    Akey string
    Bkey int
}

func main() {
    // todo struct转map（struct转json， json转map）
    demo := Demo{Akey:"a", Bkey:1}
    demoMap := make(map[string]interface{})

    elem := reflect.ValueOf(&demo).Elem() // value的list
    relType := elem.Type()  // key的list
    for i := 0; i < relType.NumField(); i++ {
        demoMap[relType.Field(i).Name] = elem.Field(i).Interface()
    }
    fmt.Println(demoMap)

}
```

二、类型断言

1、复合类型的map

如果我们需要定义一个map，但是这个map的value即有string类型，又有int类型，及其他类型该怎办？直接看代码

```
comMap := map[string]interface{}{}
comMap["Akey"] = "hello"
comMap["Bkey"] = 100
fmt.Println(comMap)
```

2、类型断言

我们定义了一个复合类型map，然后对map中的一个value进行byte转换，如下代码，运行报错，是为comT变量类型没确定，看第二个例子

```
// 运行报错
package main

import "fmt"

func main() {
    comMap := map[string]interface{}{"t":`[{"a":1},{"b":"c"}]`}
    comT := comMap["t"]
    fmt.Println([]byte(comT))
}
```

使用comT.(string)，来判断comT是否为string，是的话，ok变量为true，result指代comT变量的值result变量类型确定，可以使用[]byte(result)方法，将result转为字节了

```
// 运行通过
package main

import "fmt"

func main() {
    comMap := map[string]interface{}{"t":`[{"a":1},{"b":"c"}]`}
    comT := comMap["t"]
    result, ok := comT.(string)
    if ok{
        fmt.Println([]byte(result))
    }
}
```