**绑定请求实体到不同结构体**

通常使用 ShouldBind 绑定请求实体，底层使用的是 c.Request.Body：

package main

import (

"github.com/gin-gonic/gin"

"net/http"

)

type formA struct {

Foo string `json:"foo" xml:"foo" binding:"required"`

}

type formB struct {

Bar string `json:"bar" xml:"bar" binding:"required"`

}

func SomeHandler(c \*gin.Context) {

objA := formA{}

objB := formB{}

// c.ShouldBind 使用 c.Request.Body 并且不能被复用

if errA := c.ShouldBind(&objA); errA == nil {

c.String(http.StatusOK, `the body should be formA`)

// 这里总是会报错，因为 c.Request.Body 现在是 EOF

} else if errB := c.ShouldBind(&objB); errB == nil {

c.String(http.StatusOK, `the body should be formB`)

} else {

// 其他结构体

c.String(http.StatusOK, `the body should be other form`)

}

}

func main() {

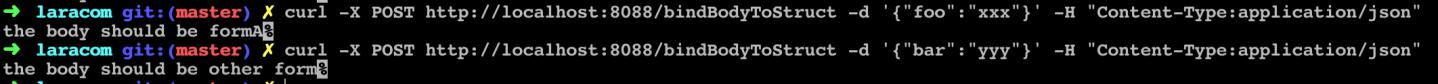
r := gin.Default()

r.POST("/bindBodyToStruct", SomeHandler)

r.Run(":8088")

}

但是 c.Request.Body 不能复用，所以上述代码运行后，访问结果如下：



要解决这个问题，可以使用 ShouldBindBodyWith 方法绑定：

func SomeHandler(c \*gin.Context) {

objA := formA{}

objB := formB{}

// c.ShouldBind 使用 c.Request.Body 并且不能被复用

if errA := c.ShouldBindBodyWith(&objA, binding.JSON); errA == nil {

c.String(http.StatusOK, `the body should be formA`)

// 这里总是会报错，因为 c.Request.Body 现在是 EOF

} else if errB := c.ShouldBindBodyWith(&objB, binding.JSON); errB == nil {

c.String(http.StatusOK, `the body should be formB`)

} else {

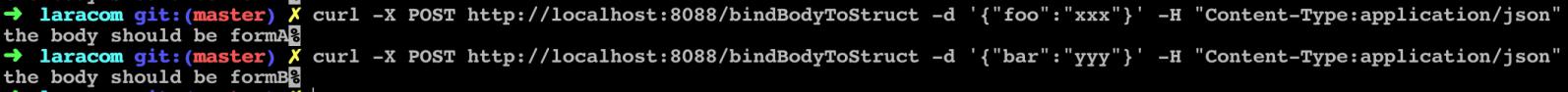
// 其他结构体

c.String(http.StatusOK, `the body should be other form`)

}

}

测试结果：



**注意事项：**

c.ShouldBindBodyWith() 会在绑定之前将请求实体存储到上下文中，这对性能有轻微的影响，所以当一次绑定可以完成功能的情况下不要用这个方法；

另外，该特性仅对这些格式有效：JSON、XML、MsgPack、ProtoBuf，对于其他格式，比如 Query、Form、FormPost、FormMultipart，可以通过多次调用 c.ShouldBind() 来完成。