链滴

Go 边看边练 -《Go 学习笔记》系列（十一）

作者：88250

原文链接：https://ld246.com/article/1438763483577

来源网站：链滴

上一篇： https://hacpai.com/article/1438699210452

ToC

5.4 表达式

根据调用者不同，方法分为两种表现形式：

instance.method(args...) ---> <type>.func(instance, args...)

前者称为 method value，后者 method expression。

两者都可像普通函数那样赋值和传参，区别在于 method value 绑定实例，而 method expression 须显式传参。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/3a844bdc99d289c1fc30a7db0ef2da6.go?embed=true" width="99%" height="600"></iframe>

需要注意，method value 会复制 receiver。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/4453dbcad943dbda38fe768a6696447.go?embed=true" width="99%" height="600"></iframe>

在汇编层面，method value 和闭包的实现方式相同，实际返回 FuncVal 类型对象。

FuncVal { method_address, receiver_copy }

可依据方法集转换 method expression，注意 receiver 类型的差异。

<iframe style="border:1px solid" src="https://wide.b3log.org/playground/4a10608a87af6746d1ac339ee3229d1.go?embed=true" width="99%" height="800"></iframe>

将方法 "还原" 成函数，就容易理解下面的代码了。

```
type Data struct{}

func (Data) TestValue() {}
func (*Data) TestPointer() {}

func main() {
    var p *Data = nil
    p.TestPointer()

    (*Data)(nil).TestPointer() // method value
    (*Data).TestPointer(nil) // method expression

    // p.TestValue() // invalid memory address or nil pointer dereference
    // (Data)(nil).TestValue() // cannot convert nil to type Data
    // Data.TestValue(nil) // cannot use nil as type Data in function argument
}
```

下一篇： https://hacpai.com/article/1438845728987
