编程语言 Go 语言 Go 编程

golang map 取值时如何选择 mapaccess 函数?

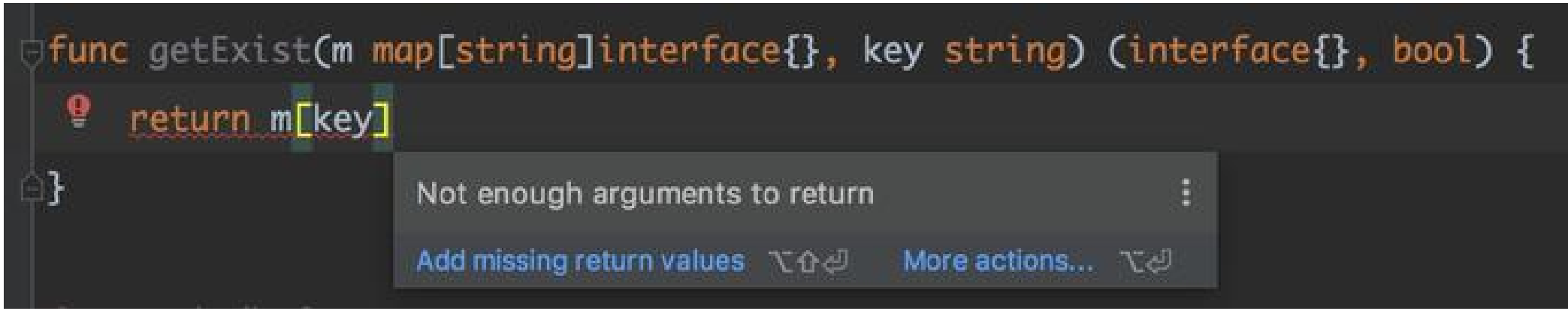
在golang 中，访问 map 的方式有两种，例子如下：

```
val := example1Map[key1]
val, ok := example1Map[key1]
```

第一种方式不判断是否存在key值，直接返回val （可能是空值）
第二种方式会返回一个bool 值，判断是否存在key 键值。（是不是和redis 的空值判断很类似）

go runtime 的 map 提供了 mapaccess1、mapaccess2、mapaccessK 三个函数来获取值，但是取值的时候是如何选择使用哪个函数的

ps：使用赋值语句可以区分 mapaccess1 和 mapaccess2，但是当在方法中使用 return 语句的时候，似乎没有办法使用 mapaccess2 方法



soolaugust
有事不回goingkeep2

在Go语言编译的类型检查期间，会根据接受参数的个数决定使用的运行时方法:

1. 当接受参数仅为一个时，会使用 `runtime.mapaccess1` ，该函数只返回一个指向目标值的指针
2. 当接受两个参数的时候，会使用 `runtime.mapaccess2` ，除了目标值，还会返回一个用于表示当前目标值是否存在的布尔值。

而 `mapaccessK` 只用于map iterator。

在Go语言中所有hash[key]及类似的操作都会转换成哈希的 `OINDEXMAP` 操作，这段代码可以在 `cmd/compile/internal/gc/walk` 中看到：

```
case OINDEXMAP:
                ...
                if n.IndexMapLValue() {
                        // This m[k] expression is on the left-hand side of an assignment.
                        ...
                } else {
                        // m[k] is not the target of an assignment.
                        fast := mapfast(t)
                        if fast == mapslow {
                                // standard version takes key by reference.
                                // order.expr made sure key is addressable.
                                key = nod(OADDR, key, nil)
                        }

                        if w := t.Elem().Width; w <= zeroValSize {
                                n = mkcall1(mapfn(mapaccess1[fast], t), types.NewPtr(t.Elem()), init, typename(t), map_, key)
                        } else {
                                z := zeroaddr(w)
                                n = mkcall1(mapfn("mapaccess1_fat", t), types.NewPtr(t.Elem()), init, typename(t), map_, key, z)
                        }
                }
                ...
```

这里调用的是 `mapaccess1` .所以无法直接在 `return` 中使用 `mapaccess2` .

发布于 2020-01-03 08:44