今日总结 2019-03-26

每日一学

问题一：

结构体类型结构一致，但 tag 不一致时，可以进行强制类型转换，tag 会被忽略。比如：

```
type Person struct {
    Name    string
    Address *struct {
        Street string
        City   string
    }
}
var data *struct {
    Name    string `json:"name"`
    Address *struct {
        Street string `json:"street"`
        City   string `json:"city"`
    } `json:"address"`
}
var person = (*Person)(data)  // ignoring tags, the underlying types are identical
```

讨论结果：

1. tag不一样就属于不同类型，但是结构一样，就可以互相转换；
2. 这里data被强转为 *Person 类型，但是原本的tag被去掉了。

面试题

问题一：

以下代码是否能正常运行？为什么？

```
package main

import (
    "fmt"
)

type Person struct {
    Name string
}

func (*Person) Study(lang string) {
    fmt.Println("study language:", lang)
}

func main() {
    var p *Person
    p.Study("Go")
```

```
}
```

讨论结果：

1. 能运行。p为person的指针类型，函数study也是*Person的方法；
2. 能正常运行。receiver 是否为 nil 不影响方法调用，而且 Study 方法中也没用到 *Person 实例；
3. 这个我试了一下，可以运行，但是如果给p.name赋值就报错了；
4. 上面的例子跟这个是一样的，结构体没有初始化，只是定义了一个类型而已。不存在指针的问题。

```go
package main

import (
    "fmt"
)

type Person struct {
    Name string
}

type a int

func (a) Study(lang string) {
    fmt.Println("study language:", lang)
}

func main() {
    var p a
    p.Study("Go")
}
```

问题二：

一到很基础的题，如果你不知道，需要加强基础学习哦。 请写出以下代码的输出。

```go
func main() {
    s := make([]int, 5)
    s = append(s, 1, 2, 3)
    fmt.Println(s)
}
```

讨论结果：

1. 输出： [0,0,0,0,0,1,2,3];
2. append是追加 ， make第二个参数是长度，int类型的默认值是0，所以是[0,0,0,0,0,1,2,3];
3. 啥场景需要提前声明slice的长度呢？球主回答：在明确知道长度的时候；
4. make() 函数的源码：`func make(t Type, size ...IntegerType) Type` 当 size 有值时，会默认给每个元素赋值(类型默认值，如：int是0，bool是false...)；new() 函数的源码：`func new(Type) *Type` ，返回Type的指针类型变量，该变量有地址，但是存储的值为空。

球主语录

- 学习编程，重要的是什么？多练、多看、多实践！跨语言学习，掌握基础语法和语言的特性之后，实战，效率来的最快！

今日链接

- 分享一个我刚开始学go遇到的很不错的边学边做练习网站 英文：https://tour.golang.org/list 中文：Go 语言之旅
-

前几天遗留问题解析

问题一：

以下代码有问题吗？为什么？

```
type student struct {
    Name string
    Age  int
}

func parseStudent() {
    m := make(map[string]*student)
    stus := []student{
        {Name: "zhou", Age: 24},
        {Name: "li", Age: 23},
        {Name: "wang", Age: 22},
    }
    for _, stu := range stus {
        m[stu.Name] = &stu
    }
}
```

解决思路：

1. 这里面关键在于 for range 循环。 在 for 循环中，stu 这个变量，看起来似乎每次都是新定义的，但实际上，Go 使用了同一块内存来保存它，我们可以通过在循环里输出 stu 的地址来验证这一点（fmt.Printf("%p\n", &stu)）；或者换一种方式，我们在 for 循环外定义 stu，即：

   ```
   var stu student
   for _, stu = range stus {}
   ```

   结果是一样的。可见，Go 重用了 这块内存。
2. 这里还得注意一点，struct 是值类型，会进行值拷贝，也就是说，stu 的内存和 stus 中 3 个元素的内存都不一样。 因此，循环中涉及到 struct 等，要特别注意此问题，最好是定义为 指针，这里也就是：stus := []*student{} 这种。
3. 题外话：学过 PHP 的人，应该熟悉，PHP 中 for 循环，如果里面使用了 引用，多次循环也会有类似的坑。
4. 补充一点，想修复，可以改为：

```go
for _, stu := stus {
      tmp := stu
      m[stu.Name] = &tmp
}

// 或

for i := 0; i < len(stus); i++ {
     m[stu.Name] = &stus[i]
}
```