随想

2021-11-23

现在是凌晨快 1 点，睡不着，稍微写写。今天白天（昨天），测试提醒我，REST 接口的输入参数应该以大写字母开头。

我一开始还紧张了一下，我 out 了？我没有按照该有的规范开发？REST 接口的参数大写开头？（接口的 `Content-Type` 是 `application/json`）

然后就去查了一下 GitHub 的 API 文档，又看了一眼 DIDs (Decentralized Identifiers) 的标准文档，以及结合长时间接触使用 json 的经验，没有啊，json 的 key，哪有大写字母开头的习惯？

后来想到了一个吐血的原因，就问了一句，“参数首字母大写，是公司的规范吗？”。测试回复的原话是：

> 我之前问过，就是你们开发自己都这么写，好像是，公有变量还是私有变量，就要首字母大写

果然是。（微笑）

权限控制

Java 里的权限修饰符是很基础的概念，用来控制类的访问权限，一共有 3 个关键词、4 种情况，比如 `public` 表示公开，`private` 表示私有：

```
public class aaa {
    public int a;
    private String b;
}
```

具体内容可以参考 Java 的文档： Controlling Access to Members of a Class

像 Python 那样的脚本语言本身没有访问权限控制的概念，所以约定俗成地认为以 _ 开头命名的变量是私有的，不过没有代码层面的保护，只是命名上的区别。

```
class aaa:
    a = 1
    _b = 2
```

Pthon 里 __ 开头的变量名可以达到代码层面私有变量的效果，但把 __ 用做私有变量并不完全是 Python 的本意。参考 Pthon 的文档：9.6. Private Variables

众所周知，Golang 是一种奇葩的语言，不但要求你大括号必须写在函数名同一行，还会要求你必须做一些什么事情，比如，大写字母开头的变量意味着公开，小写字母开头的变量意味着私有。

Golang 对变量大小写的区分，可不仅仅是命名上的区别，而是代码层面的控制。下面的代码会输出什么？

```
func main() {
    a := struct {
        i int
    }{
        i: 1,
    }
    s, _ := json.Marshal(a)
    fmt.Println(string(s))
}
```

当然是 `{}`，由于 `i` 是小写的，在 marshal 的时候自动忽略了。

这其实算是比较不正常的要求了。Golang 为了设计上的 “简洁”，很大程度上减少关键字、关键字复用，但是又不得不提供某些能力，就需要一些奇怪的方式做变通。

我在上面的 Java 代码中，刻意用了小写的 `aaa` 做类名。Java 的类一般使用大写字母开头，但是不会在代码层面限制你把代码写成什么样子。

虽然我工作使用 Golang，但我不是它的教徒，当然，也不是其他语言的教徒。如果你仍然是某种语言的教徒，祝愿你可以尽早改变一下观念。

所以，由于 Golang 的一些 “特性”，为了在序列化的时候方便一点，项目的 REST 接口就习惯于使用大写字母开头的变量名作为 key 了。

睡觉了，明天再写。

没有问题

json 的 key 大写或者小写开头，其实都没有问题。输出参数为了省去写 `` `json` `` 的麻烦，就默认大写开头，所以输入参数也大写开头了。这是一种工程上的对称，并没有什么问题。

想起之前有一次，我把 REST 接口设计为，输入参数用驼峰命名 `myName`，输出参数用下划线分割 `my_name`。这样做的原因，是参数输入的时候，Java 的 Entity 能够把 POST Body 里驼峰命名的变量自动识别到对象上。返回参数时，程序直接把从数据库查出来的数据返回出去，是下划线形式的（不符合开发规范是真的）。这样做可以达到代码上的最简洁，但确实输入输出不对称不合适。

一件事

如果只是这么点事，就不值得写什么了。昨天测试说我什么地方该怎么样做的时候，我想到另一件事，在之前的公司里发生的事。

URL 的输入参数有两种形式，一种是 `.com/?a=1&b=2`，另一种是 `.com/:a/:b`。当时的项目里全是第二种形式，也就是路由形式的参数。

全部是第二种形式，有时候特别长，比如 `.com/:a/:b/:c/:d/:e`。参数很多时候是查询条件，如果中间的某个参数不需要作为条件，也就是不需要传参数，但是这种路由形式的参数又不能跳过中间的一段，就用了一种很奇葩的方式变通：`all`。用 `all` 来代替不需要传参的情况，`.com/:a/:b/all/:d/:e`。程序里面的判断也是各种离谱。

REST 接口本身是一种 Web 服务，但是如果缺少 Web 开发的基础，就容易做的有一些欠缺。

我当时怀疑开发那个项目的人技术水平有问题，也就是我当时名义上的小组长。好吧，也许是一段不怎么开心的往事。

顺便解释一下，我也不是有事没事就会对别人有意见。我当时所在的公司，有一种绩效的制度，每个季度开始前都需要写绩效计划，大概是接下来一段时间要做什么之类，一般是组长制定一个大方向上的目标，然后组员分配分别完成一些节点。我因为质疑他的水平，担心他制定的目标比较低，限制我的发挥，就借题发挥试探一下。对别人有意见对我自己没有任何好处，我也是出于一种恐惧，担心自己接下来会置于难以发挥的处境。

我当时并没有想离职，所以还是在极力想办法改善自己周围的环境和情况，给自己争取一个自己可以适应的状况。记得当时有其他同事离职的时候，（以前好像说过？好像说过，不记得了，再说一遍），我开玩笑说，工作中最开心的时候，可能就是提离职的时候和收到 offer 的时候了。同事说，最开心的时候是年底发奖金的时候……也是开玩笑。后来接着说，如果不是到没有办法的地步，不会有人愿意离职的。

我知道参数形式这种问题是小问题，但是我坚持认为，在当时的接口设计上，应该用参数形式的

参数而不是路由形式的参数。参数形式的参数明明更好用，为什么不用？说说你的理由，我已经拿出了话题，还把话头递给了你，你说一说啊，你的理由，除了“以前的人就是这么写的”之外，你有没有其他的在技术方面的思考？能不能够在技术方面碾压我，以小见大，体现一下自己的技术能力？

我的基本观点是，他缺少 Web 开发的经验，不是认为路由形式的参数好用，而是完全不知道有参数形式的参数存在。我分析有两种可能，一种是他自己有实力，但是不轻易和我说，另一种就是自己没有实力。所以啊，我从一个很小的问题，制造了话题，给他说话的机会，甚至之后用比较强硬的态度在 leader 面前，表达了我对他能力的质疑。我心想，我都这么挑事了，要真有实力，总该表现出来了吧。

唉再之后就是一些收场环节了。

设身处地

我当时也想过，如果我是他，我会怎么办？

如果以后有小伙子那样对我，我能处理地比他更好吗？

如果有人质疑我的技术实力，或者揪着我的失误不放，借题发挥，我该怎么办？

我也知道，我好像破坏了一种和谐的宁静。

然后呢

现在的我，已经没有兴趣在类似那样的小问题上较真了，怎么样是对的，什么样是错的，有什么所谓呢。

不过过去的经历好像也是必要的。如果不是过去认真过，可能现在也还是会好奇：如果在一些问题上，我坚持自己的观点，或者非要争个对错，会是什么样的结果呢。

由于过去的经历，我已经知道是什么样的结果了。

其实结果不太好，也有点无聊。

我不太喜欢也并不期望是那样。

暂时没有然后了。