老外总结的14条Go接口最佳实践，有些不一样

==========

最近几个月，没事喜欢看看老外写的技术文章，发现他们的一些思考维度真的有些不太一样。当然，他们写的文章大多数没有国内的那么卷。

今天这篇文章是关于Go语言中接口设计的一些最佳实践，与Java等语言不尽相似，但又带着Go语言的特色，可以对照学习，拓展编程思想层面的认知面。

以下是在Go中使用接口的一些最佳实践：

1、**优先小接口**：接口是Go中的强大工具，但要保持它们小巧并专注于特定任务。这样有助于提高可读性、可维护性，并减少耦合。

2、**避免类型切换**：类型切换会增加代码的复杂度且难以维护，要优先使用小巧且专注的接口，而不是类型切换。

3、**使用接口进行模拟测试**：接口是进行模拟测试的好工具，因为它们允许将真实实现替换为用于测试目的的模拟实现。

4、**使用组合**：使用组合来实现多个接口，而不是将一个接口嵌入到另一个接口中。这有助于提高可维护性并减少耦合。

PS：在Go中，组合的使用非常普遍，通过组合可以实现结构体（对照Java类）的继承关系，而接口的组合同样可对照Java接口的继承关系。

5、**选择正确的抽象级别**：在定义接口时，请考虑抽象级别，并确保接口既不太具体也不太通用（泛化）。

6、**避免空接口**：空接口没有方法，可以接受任何类型，因此除非确实需要它们提供的灵活性，否则请避免使用。

PS：空接口的存在，与Java中Object的存在位置类型，当参数定义为空接口时，可接收任何参数。就好像你把参数定义为Object一样。

7、**使用类型断言**：类型断言用于断言接口值的底层类型，但应谨慎使用，因为如果类型断言不正确，它们可能会导致恐慌（panic）。

PS：panic经常被翻译为恐慌。在Golang中，panic是一种表示严重错误的异常情况。当程序遇到无法处理的错误时，它会引发panic，并中断当前的执行流程。panic类似于Java中的RuntimeException，它们都是属于运行时异常，并且不需要显式地在代码中声明或捕获。

但是，与Java中的受检异常（Checked Exception）不同，Golang中的panic是一种不可恢复的异常，它会导致程序崩溃并触发执行堆栈的展开。因此，在Golang中，通常建议使用panic来处理程序无法恢复的错误，而不是像Java中那样使用受检异常来表示可预期的错误情况。

8、**完全实现接口**：在实现接口时，请确保实现接口中定义的所有方法，否则实现将无法编译。

9、**使用接口声明行为**：Go中的接口用于声明对象的行为，而不是其实现。使用接口为多种类型定义通用行为。

10、**对于具体类型使用类型断言**：在必要时，使用类型断言访问接口值的具体类型。

11、**使用空接口**：空接口（interface {}）是一种通用类型，可以容纳任何类型的值。谨慎使用它，因为它可能使代码更难以理解。

12、**避免转换函数**：避免编写将类型转换为接口的显式转换函数。这通常会导致代码可读性较差且难以维护。

13、**使用接口组合**：使用接口组合来声明实现多个接口的类型。当想要将多种行为组合到一个类型中时，这非常有用。

14、**避免隐藏依赖**：在使用接口时，请注意隐藏依赖。在将其实现为类型之前，要了解接口所需的所有方法集合。

原文链接: https://juejin.cn/post/7296771769065734171