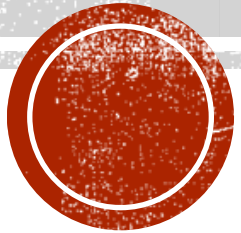
# 11.29技术分享

刘子杰

Go语言基本概念介绍



# CONTEXT

- 1、Go语言基本特性
- 2、Go语言的主要语法介绍
- 3、Post a sample

# GO语言特性

- 1、Go既有可以媲美C、Cpp的执行速度，又可以像RubyPython一样进行快速的开发。
- 2、Go的编译只会关注哪些被引用的库，而像是Java、C、C++则会依赖链中所有的库，因此Go的编译速度更快，多数的Go程序可以在1秒内编译完成，而就算是编译整个Go语言的源码树也只需要大概20秒左右的时间。
- 3、现代的计算机一般都拥有多核，但是大部分的编程语言都没有有效的工具使程序可以轻易的利用这些资源。这些语言需要写大量的线程同步代码来利用多个核，容易导致错误。Go语言可以通过内置的goroutine和channel来更好的支持并发。
- 4、通道可以帮助用户避免共享内存的访问问题，它保证了同一时刻只会有一个goroutine的修改数据。通道在两个goroutine之间传输数据是同步的。
- 5、Go语言提供了灵活的、无继承的类型系统。Go语言采用了一种被称为组合(composition)的设计模式，不同于传统的面向对象的语言是通过继承的树状形态来实现扩展结构，但是Go语言的扩展结构是通过构建的更小的类型来组合成更大的类型。

- 6、python和go语言相比，go语言相对较新，go语言的库没有python的丰富，毕竟python发展了几十年，go语言最大的优点是协程，速度快，支持高并发，但任何支撑高并发的语言编程都不是一件易事，最大的缺点是，名称有google的go，所有很多其他的巨头公司本着竞争的观念对有类似商标的语言有抵制心理。而python是开源，用得公司特别多。go语言的语法如果能像c#，java类似，可能会发展得更好。很多语言为什么会用它，不在于语法，不在于性能，主要在于这个语言开发了什么成功的产品，另人会mock这个产品，用这个语言，Docker就是go语言开发的，适就于大规模分发容器。

# GO语言的主要语法介绍

- 1、切片Slice
- Slice 是 Go 中一个关键的数据类型，是一个比数组更加强大的序列接口
- 不像数组，slice 的类型仅由它所包含的元素决定（不像数组中还需要元素的个数）。要创建一个长度非零的空slice，需要使用内建的方法 make。这里我们创建了一个长度为3的 string 类型 slice（初始化为零值）。我们可以和数组一样设置和得到值如你所料，len 返回 slice 的长度作为基本操作的补充，slice 支持比数组更多的操作。其中一个是内建的 append，它返回一个包含了一个或者多个新值的 slice。注意我们接受返回由 append返回的新的 slice 值。Slice 也可以被 copy。这里我们创建一个空的和 s 有相同长度的 slice c，并且将 s 复制给 c。Slice 支持通过 slice[low:high] 语法进行“切片”操作。例如，这里得到一个包含元素 s[2], s[3],s[4] 的 slice。这个 slice 从 s[0] 到（但是包含）s[5]。这个 slice 从（包含）s[2] 到 slice 的后一个值。我们可以在一行代码中声明并初始化一个 slice 变量。Slice 可以组成多维数据结构。内部的 slice 长度可以不同，这和多位数组不同。

# 2、闭包

- Go 支持通过 闭包来使用 匿名函数。匿名函数在你想定义一个不需要命名的内联函数时是很实用的。这个 **intSeq** 函数返回另一个在 **intSeq** 函数体内定义的匿名函数。这个返回的函数使用闭包的方式 隐藏 变量 **i**。
- 我们调用 **intSeq** 函数，将返回值（也是一个函数）赋给**nextInt**。这个函数的值包含了自己的值 **i**，这样在每次调用 **nextInt** 时都会更新 **i** 的值。通过多次调用 **nextInt** 来看看闭包的效果。为了确认这个状态对于这个特定的函数是唯一的，我们重新创建并测试一下。

# 3、 INTERFACE

- 接口 是方法特征的命名集合。要在 **Go** 中实现一个接口，我们只需要实现接口中的所有方法。这里我们让 **rect** 实现了 **geometry** 接口。如果一个变量的是接口类型，那么我们可以调用这个被命名的接口中的方法。这里有一个一通用的 **measure** 函数，利用这个特性，它可以用在任何 **geometry** 上。结构体类型 **circle** 和 **rect** 都实现了 **geometry** 接口，所以我们可以使用它们的实例作为 **measure** 的参数。

# 4、协程

- Go 协程 在执行上来说是轻量级的线程。
-

  假设我们有一个函数叫做 f(s)。我们使用一般的方式调并同时运行。使用 go f(s) 在一个 Go 协程中调用这个函数。这个新的 Go 协程将会并行的执行这个函数调用。你也可以为匿名函数启动一个 Go 协程。现在这两个 Go 协程在独立的 Go 协程中异步的运行，所以我们需要等它们执行结束。这里的 Scanln 代码需要我们在程序退出前按下任意键结束。当我们运行这个程序时，将首先看到阻塞式调用的输出，然后是两个 Go 协程的交替输出。这种交替的情况表示 Go 运行时是以异步的方式运行协程的。

# 5、CHANEL

- 通道 是连接多个 Go 协程的管道。你可以从一个 Go 协程将值发送到通道，然后在别的 Go 协程中接收。使用 make(chan val-type) 创建一个新的通道。通道类型就是他们需要传递值的类型。使用 channel <- 语法 发送 一个新的值到通道中。这里我们在一个新的 Go 协程中发送 "ping" 到上面创建的messages 通道中。使用 <-channel 语法从通道中 接收 一个值。这里将接收我们在上面发送的 "ping" 消息并打印出来。我们运行程序时，通过通道，消息 "ping" 成功的从一个 Go 协程传到另一个中。默认发送和接收操作是阻塞的，直到发送方和接收方都准备完毕。这个特性允许我们，不使用任何其它的同步操作，来在程序结尾等待消息 "ping"。

# GIVE A SAMPLE

- Just look through the code given in the repository

# THANKS FOR WATCHING