|  |
| --- |
| <http://blog.51cto.com/speakingbaicai/1703249> |
|  |

接口是一个自定义类型，它是一组方法的集合。接口两个特点：1、接口本质是一种自定义类型，因此不要将golang中的接口简单的理解未c++/java中的接口，后者仅用于声明方法签名 2、接口是一种特殊的自定义类型，其中没有数据成员，只有方法（也可以为空）

接口是完全抽象的，因此不能将其实例化。然而，可以创建一个其类型未接口的变量，它可以被赋值为任何满足该接口类型的实际类型的值。

接口的特性：

1. 只有某个类型实现了接口的方法，那么我们说该类型实现了此接口。该类型的值可以赋值给该接口的变量。
2. 任何类型的值都可以赋值给空接口 interface{}

接口的特性是golang支持鸭子类型的基础，即如果它走起来像鸭子，叫起来像鸭子（实现了接口里的方法），它就是一只鸭子（可以被赋值给接口的值）。由此golang提供了一种游离于 类、继承、模版之外更加灵活强大的选择。

|  |
| --- |
| type Exchanger interface {         exchange()  }    type StringPair struct {         first, second string  }    type Point[2]int    func (sp \*StringPair) exchange() {         sp.first, sp.second = sp.second, sp.first  }    func (p \*Point) exchange() {         p[0], p[1] = p[1], p[0]  }    func exchangeThese(exchangers ...Exchanger) {         for \_, exchanger := range exchangers {                exchanger.exchange()         }  }    func main() {         pair1 := StringPair{"abc","def"}         pair2 := StringPair{"ghi","jkl"}         point := Point{5, 7}           fmt.Println(pair1, pair2, point)         pair1.exchange()         pair2.exchange()         point.exchange()         fmt.Println(pair1, pair2, point)           // exchangeThese(pair1, pair2) //wrong         exchangeThese(&pair1, &pair2)         fmt.Println(pair1, pair2)  } |
| wKioL1YfYriDNBSMAACXshVCpYE028.jpg |

上述例子中，自定义类型StringPair 和 Point 指针实现了接口Exchanger所需的方法，因此该类型的值可以被赋值给接口的值。