链滴

GOlang 对象排序

作者：K
原文链接：https://ld246.com/article/1539672131843
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

golang sort 包

包中这样介绍.

type Interface

```
type Interface interface {
    // Len方法返回集合中的元素个数
    Len() int
    // Less方法报告索引i的元素是否比索引j的元素小
    Less(i, j int) bool
    // Swap方法交换索引i和j的两个元素
    Swap(i, j int)
}
```

一个满足sort.Interface接口的（集合）类型可以被本包的函数进行排序。方法要求集合中的元素可以整数索引。

golang 中排序只需要重写 len, less, swap 方法

func Sort

```
func Sort(data Interface)
```

它调用1次data.Len确定长度，调用O(n*log(n))次data.Less和data.Swap。本函数不能保证排序的稳性（即不保证相等元素的相对次序不变）。

使用

结构体

```
type SortJson struct {
   Name string `json:"name"`
   Count int `json:"count"`
}
```

重写 Len(), Swap方法

```
type SortJsonSlice []SortJson

func (s SortJsonSlice) Len() int {
   return len(s)
}

func (s SortJsonSlice) Swap(i, j int) {
   s[i], s[j] = s[j], s[i]
}

func (s SortJsonSlice) Less(i, j int) bool {
   return s[i].Count > s[j].Count
}
```

test

```
func TestSortObject(t *testing.T) {
   dj := SortJsonSlice{SortJson{Name:"a",Count:1},{Name:"c",Count:3},{Name:"b",Count:2},{Na
e:"f",Count:6},{Name:"e",Count:4},{Name:"g",Count:7}}
   fmt.Println(dj)
   sort.Sort(dj)
   fmt.Println(dj)
}
```

这个打印的结果就是

```
=== RUN   TestSortObject
[{a 1} {c 3} {b 2} {f 6} {e 4} {g 7}]
[{g 7} {f 6} {e 4} {c 3} {b 2} {a 1}]
--- PASS: TestSortObject (0.00s)
```

排序成功

升级使用

上面的写就已经很方便了, 但是有个问题. 如果你可能需要多种排序方式(正序, 倒序, 时间, 数量). 就会烦.

可以在次重写上面重写过的数据 只重写 Less 方法.

比如, 根据count正序. (上面的方法是倒序)

```
// 将 Less 绑定到 OrderCount 上
type OrderCount struct{
   SortJsonSlice
```

```
}

// 将 DeviceJsonSlice 包装起来到 ByWeight 中
func (s OrderCount) Less(i, j int) bool {
   return s.SortJsonSlice[i].Count < s.SortJsonSlice[j].Count
}
```

如果要添加一个排序方式 根据 Name 排序:

```
type OrderName struct{
   SortJsonSlice
}

// 将 DeviceJsonSlice 包装起来到 ByWeight 中
func (s OrderCount) Less(i, j int) bool {
   return s.SortJsonSlice[i].Name < s.SortJsonSlice[j].Name
}
```

这样只要给每种排序方式重写一个 Less 方法就好了.

test

```
func TestSortObject(t *testing.T) {
   dj := OrderCount{[]DeviceJson{{Name:"a",Count:1},{Name:"c",Count:3},{Name:"b",Count:2},{
ame:"f",Count:6},{Name:"e",Count:4},{Name:"g",Count:7}}}
   fmt.Println(dj)
   sort.Sort(dj)
   fmt.Println(dj)
}
```

```
=== RUN   TestSortObject
{[{a 1} {c 3} {b 2} {f 6} {e 4} {g 7}]}
{[{a 1} {b 2} {c 3} {e 4} {f 6} {g 7}]}
--- PASS: TestSortObject (0.00s)
PASS
```