-

Content Menu

-
-
- Push*
- Pop*
- Move*/Insert*
- Join
- Remove*
- JSON/


```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/container/glist"
)

func main() {
        // Not concurrent-safe in default.
        l := glist.New()

    // Push
        l.PushBack(1) //
    l.PushBack(2) //
    e := l.PushFront(0) //

    // Insert
        l.InsertBefore(e, -1) //0
    l.InsertAfter(e, "a") //0
        fmt.Println(l)

        // Pop Pop list
    fmt.Println(l.PopFront()) //
    fmt.Println(l.PopBack()) //
    fmt.Println(l)

        // All
        fmt.Println(l.FrontAll()) //
    fmt.Println(l.BackAll())  //

    // Output:
        // [-1,0,"a",1,2]
        // -1
        // 2
        // [0,"a",1]
        // [0 "a" 1]
        // [1 "a" 0]
}
```

RLockFuncLockFunc

```
package main

import (
        "container/list"
        "fmt"
        "github.com/gogf/gf/v2/container/garray"
        "github.com/gogf/gf/v2/container/glist"
)

func main() {
        // concurrent-safe list.
        l := glist.NewFrom(garray.NewArrayRange(1, 9, 1).Slice(), true)
    fmt.Println(l)
        // iterate reading from head.
        l.RLockFunc(func(list *list.List) {
                length := list.Len()
                if length > 0 {
                        for i, e := 0, list.Front(); i < length; i, e =
i+1, e.Next() {
                                fmt.Print(e.Value)
```

```
                        }
                }
        })
        fmt.Println()
        // iterate reading from tail.
        l.RLockFunc(func(list *list.List) {
                length := list.Len()
                if length > 0 {
                        for i, e := 0, list.Back(); i < length; i, e =
i+1, e.Prev() {
                                fmt.Print(e.Value)
                        }
                }
        })

    fmt.Println()

    // iterate reading from head using IteratorAsc.
        l.IteratorAsc(func(e *glist.Element) bool {
                fmt.Print(e.Value)
                return true
        })
        fmt.Println()
        // iterate reading from tail using IteratorDesc.
        l.IteratorDesc(func(e *glist.Element) bool {
                fmt.Print(e.Value)
                return true
        })

        fmt.Println()

        // iterate writing from head.
        l.LockFunc(func(list *list.List) {
                length := list.Len()
                if length > 0 {
                        for i, e := 0, list.Front(); i < length; i, e =
i+1, e.Next() {
                                if e.Value == 6 {
                                        e.Value = "M"
                                        break
                                }
                        }
                }
        })
        fmt.Println(l)

    // Output:
        // [1,2,3,4,5,6,7,8,9]
        // 123456789
        // 987654321
        // 123456789
        // 987654321
        // [1,2,3,4,5,M,7,8,9]
```

Push*

```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/container/glist"
        "github.com/gogf/gf/v2/frame/g"
)

func main() {
        l := glist.NewFrom(g.Slice{1, 2, 3, 4, 5})

        l.PushBack(6)
        fmt.Println(l)

        l.PushFront(0)
        fmt.Println(l)

    //
        l.PushBacks(g.Slice{7, 8})
        fmt.Println(l)

    //
        l.PushFronts(g.Slice{-1, -2})
        fmt.Println(l)

        l.PushFrontList(glist.NewFrom(g.Slice{"a", "b", "c"}))
        l.PushBackList(glist.NewFrom(g.Slice{"d", "e", "f"}))
        fmt.Println(l)

        // Output:
        // [1,2,3,4,5,6]
        // [0,1,2,3,4,5,6]
        // [0,1,2,3,4,5,6,7,8]
        // [-2,-1,0,1,2,3,4,5,6,7,8]
        // ["a","b","c",-2,-1,0,1,2,3,4,5,6,7,8,"d","e","f"]

}
```

Pop*

```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/container/glist"
        "github.com/gogf/gf/v2/frame/g"
)

func main() {
        l := glist.NewFrom(g.Slice{1, 2, 3, 4, 5, 6, 7, 8, 9})

        fmt.Println(l.PopBack())
        fmt.Println(l.PopBacks(2))
        fmt.Println(l.PopFront())
        fmt.Println(l.PopFronts(2))

        fmt.Println(glist.NewFrom(g.Slice{"a", "b", "c", "d"}).
PopFrontAll())
        fmt.Println(glist.NewFrom(g.Slice{"a", "b", "c", "d"}).
PopBackAll())

        // Output:
        // 9
        // [8 7]
        // 1
        // [2 3]
    // [4,5,6]
        // [a b c d]
        // [d c b a]
}
```

Move*/Insert*

```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/container/glist"
        "github.com/gogf/gf/v2/frame/g"
)

func main() {
        l := glist.NewFrom(g.Slice{1, 2, 3, 4, 5, 6, 7, 8, 9})

        l.MoveToBack(l.Front()) //(1)  [2,3,4,5,6,7,8,9,1]
    l.MoveToFront(l.Back().Prev()) //(9)  [9,2,3,4,5,6,7,8,1]
    fmt.Println(l)

        // 2
        l.MoveBefore(l.Front().Next(), l.Front())
        // 8
        l.MoveAfter(l.Back().Prev(), l.Back())
        fmt.Println(l)

        //
        l.InsertBefore(l.Back(), "a")
        //
        l.InsertAfter(l.Front(), "b")

        // Output:
        // [9,2,3,4,5,6,7,8,1]
        // [2,9,3,4,5,6,7,1,8]
        // [2,"b",9,3,4,5,6,7,1,"a",8]
}
```

Join

```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/container/glist"
        "github.com/gogf/gf/v2/frame/g"
)

func main() {
        var l glist.List
        l.PushBacks(g.Slice{"a", "b", "c", "d"})

        fmt.Println(l.Join(","))

        // Output:
        // a,b,c,d
}
```

Remove*

```
package main

import (
        "fmt"
        "github.com/gogf/gf/v2/container/glist"
        "github.com/gogf/gf/v2/frame/g"
)

func main() {
    l := glist.NewFrom(g.Slice{0, 1, 2, 3, 4, 5, 6, 7, 8, 9})
        fmt.Println(l)

        fmt.Println(l.Remove(l.Front()))
        fmt.Println(l)

        l.Removes([]*glist.Element{l.Front(), l.Front().Next()})
        fmt.Println(l)

        l.RemoveAll()
        fmt.Println(l)

        // Output:
        // [0,1,2,3,4,5,6,7,8,9]
        // 0
        // [1,2,3,4,5,6,7,8,9]
        // [3,4,5,6,7,8,9]
        // [] }
```

JSON/

glistjson/

- Marshal

```
package main

import (
    "encoding/json"
    "fmt"
    "github.com/gogf/gf/v2/container/glist"
    "github.com/gogf/gf/v2/frame/g"
)

func main() {
    type Student struct {
        Id     int
        Name   string
        Scores *glist.List
    }
    s := Student{
        Id:     1,
        Name:   "john",
        Scores: glist.NewFrom(g.Slice{100, 99, 98}),
    }
    b, _ := json.Marshal(s)
    fmt.Println(string(b))

        // Output:
        // {"Id":1,"Name":"john","Scores":[100,99,98]}
}
```

- Unmarshal

```
package main


import (
    "encoding/json"
    "fmt"
    "github.com/gogf/gf/v2/container/glist"
)


func main() {
    b := []byte(`{"Id":1,"Name":"john","Scores":[100,99,98]}`)
    type Student struct {
        Id     int
        Name   string
        Scores *glist.List
    }
    s := Student{}
    json.Unmarshal(b, &s)
    fmt.Println(s)

        // Output:
        // {1 john [100,99,98]}
}
```