-

https://pkg.go.dev/github.com/gogf/gf/v2/container/gring

Content Menu

- New
- Val
- Len
- Cap
- Set
- Put
- Move
- Prev
- Next
- Link
- Unlink
- RLockIteratorNext
- RLockIteratorPrev
- SliceNext
- SlicePrev

New

- Newcapsafefalse
-

```
New(cap int, safe ...bool) *Ring
```

-

```
func ExampleNew() {
        // Non concurrent safety
        gring.New(10)

        // Concurrent safety
        gring.New(10, true)

        // Output:
}
```

Val

- Val
-

```
Val() interface{}
```

-

```
func ExampleRing_Val() {
        r := gring.New(10)
        r.Set(1)
        fmt.Println("Val:", r.Val())

        r.Next().Set("GoFrame")
        fmt.Println("Val:", r.Val())

        // Output:
        // Val: 1
        // Val: GoFrame
}
```

Len

- LenRing
-

```
Len() int
```

-

```
func ExampleRing_Len() {
        r1 := gring.New(10)
        for i := 0; i < 5; i++ {
                r1.Set(i).Next()
        }
        fmt.Println("Len:", r1.Len())

        r2 := gring.New(10, true)
        for i := 0; i < 10; i++ {
                r2.Set(i).Next()
        }
        fmt.Println("Len:", r2.Len())

        // Output:
        // Len: 5
        // Len: 10
}
```

Cap

- CapRing
-

```
Cap() int
```

-

```
func ExampleRing_Cap() {
        r1 := gring.New(10)
        for i := 0; i < 5; i++ {
                r1.Set(i).Next()
        }
        fmt.Println("Cap:", r1.Cap())

        r2 := gring.New(10, true)
        for i := 0; i < 10; i++ {
                r2.Set(i).Next()
        }
        fmt.Println("Cap:", r2.Cap())

        // Output:
        // Cap: 10
        // Cap: 10
}
```

Set

- SetValue
-

```
Set(value interface{}) *Ring
```

-

```
func ExampleRing_Set() {
        r := gring.New(10)
        r.Set(1)
        fmt.Println("Val:", r.Val())

        r.Next().Set("GoFrame")
        fmt.Println("Val:", r.Val())

        // Output:
        // Val: 1
        // Val: GoFrame
}
```

Put

- PutValuering
-

```
Put(value interface{}) *Ring
```

-

```
func ExampleRing_Put() {
        r := gring.New(10)
        r.Put(1)
        fmt.Println("Val:", r.Val())
        fmt.Println("Val:", r.Prev().Val())

        // Output:
        // Val: <nil>
        // Val: 1
}
```

Move

- n % r.Len()n>=0
-

```
Move(n int) *Ring
```

-

```
func ExampleRing_Move() {
        r := gring.New(10)
        for i := 0; i < 10; i++ {
                r.Set(i).Next()
        }
        // ring at Pos 0
        fmt.Println("CurVal:", r.Val())

        r.Move(5)

        // ring at Pos 5
        fmt.Println("CurVal:", r.Val())

        // Output:
        // CurVal: 0
        // CurVal: 5
}
```

Prev

- Prevring
-

```
Prev() *Ring
```

-

```
func ExampleRing_Prev() {
        r := gring.New(10)
        for i := 0; i < 5; i++ {
                r.Set(i).Next()
        }

        fmt.Println("Prev:", r.Prev().Val())
        fmt.Println("Prev:", r.Prev().Val())

        // Output:
        // Prev: 4
        // Prev: 3
}
```

Next

- Next ring
-

```
Next() *Ring
```

-

```
func ExampleRing_Next() {
        r := gring.New(10)
        for i := 5; i > 0; i-- {
                r.Set(i).Prev()
        }

        fmt.Println("Prev:", r.Next().Val())
        fmt.Println("Prev:", r.Next().Val())

        // Output:
        // Prev: 1
        // Prev: 2
}
```

Link

-
- 1. Linkring rring sr.Next()sr.Next()rLinkring rLenCaprsLenCap
- 2. rsringringrsringringr.Next()nil
- 3. rsringringsrs
-

```
(r *Ring) Link(s *Ring) *Ring
```

-

```
func ExampleRing_Link_Common() {
        r := gring.New(10)
        for i := 0; i < 5; i++ {
                r.Set(i).Next()
        }

        s := gring.New(10)
        for i := 0; i < 10; i++ {
                val := i + 5
                s.Set(val).Next()
        }

        r.Link(s) // Link Ring s to Ring r

        fmt.Println("Len:", r.Len())
        fmt.Println("Cap:", r.Cap())
        fmt.Println(r.SlicePrev())
        fmt.Println(r.SliceNext())

        // Output:
        // Len: 15
        // Cap: 20
        // [4 3 2 1 0]
        // [5 6 7 8 9 10 11 12 13 14]
}
```

```
func ExampleRing_Link_SameRing() {
        r := gring.New(10)
        for i := 0; i < 5; i++ {
                r.Set(i).Next()
        }

        same_r := r.Link(r.Prev())

        fmt.Println("Len:", same_r.Len())
        fmt.Println("Cap:", same_r.Cap())
        fmt.Println(same_r.SlicePrev())
        fmt.Println(same_r.SliceNext())

        // Output:
        // Len: 1
        // Cap: 1
        // [4]
        // [4]
}
```

Unlink

- `Unlinkr.next()ringn % r.len()n % r.len() == 0ringsubring`
-

```
Set(value interface{}) *Ring
```

-

```
func ExampleRing_Unlink() {
        r := gring.New(10)
        for i := 0; i < 10; i++ {
                r.Set(i).Next()
        }

        fmt.Println("Before Unlink, Len:", r.Len())
        fmt.Println("Before Unlink, Cap:", r.Cap())
        fmt.Println("Before Unlink, ", r.SlicePrev())
        fmt.Println("Before Unlink, ", r.SliceNext())

        r.Unlink(7)

        fmt.Println("After Unlink, Len:", r.Len())
        fmt.Println("After Unlink, Cap:", r.Cap())
        fmt.Println("After Unlink, ", r.SlicePrev())
        fmt.Println("After Unlink, ", r.SliceNext())

        // Output:
        // Before Unlink, Len: 10
        // Before Unlink, Cap: 10
        // Before Unlink,  [0 9 8 7 6 5 4 3 2 1]
        // Before Unlink,  [0 1 2 3 4 5 6 7 8 9]
        // After Unlink, Len: 7
        // After Unlink, Cap: 7
        // After Unlink,  [1 7 6 5 4 3 2]
        // After Unlink,  [1 2 3 4 5 6 7]
}
```

RLockIteratorNext

- RLockIteratorNextRWMutex.RLockfunc ffun ftruefalse
-

```
RLockIteratorNext(f func(value interface{}) bool)
```

-

```
func ExampleRing_RLockIteratorNext() {
        r := gring.New(10)
        for i := 0; i < 10; i++ {
                r.Set(i).Next()
        }

        r.RLockIteratorNext(func(value interface{}) bool {
                if value.(int) < 5 {
                        fmt.Println("IteratorNext Success, Value:",
value)
                        return true
                }

                return false
        })

        // Output:
        // IteratorNext Success, Value: 0
        // IteratorNext Success, Value: 1
        // IteratorNext Success, Value: 2
        // IteratorNext Success, Value: 3
        // IteratorNext Success, Value: 4
}
```

RLockIteratorPrev

- RLockIteratorPrevRWMutex.RLockfunc ffun ftruefalse
-

```
RLockIteratorPrev(f func(value interface{}) bool)
```

-

```
func ExampleRing_RLockIteratorPrev() {
        r := gring.New(10)
        for i := 0; i < 10; i++ {
                r.Set(i).Next()
        }

        // move r to pos 9
        r.Prev()

        r.RLockIteratorPrev(func(value interface{}) bool {
                if value.(int) >= 5 {
                        fmt.Println("IteratorPrev Success, Value:",
value)
                        return true
                }

                return false
        })

        // Output:
        // IteratorPrev Success, Value: 9
        // IteratorPrev Success, Value: 8
        // IteratorPrev Success, Value: 7
        // IteratorPrev Success, Value: 6
        // IteratorPrev Success, Value: 5
}
```

SliceNext

- SliceNextslice
-

```
SliceNext() []interface{}
```

-

```
func ExampleRing_SliceNext() {
        r := gring.New(10)
        for i := 0; i < 10; i++ {
                r.Set(i).Next()
        }

        fmt.Println(r.SliceNext())

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

SlicePrev

- SlicePrevslice
-

```
SlicePrev() []interface{}
```

-

```
func ExampleRing_SlicePrev() {
        r := gring.New(10)
        for i := 0; i < 10; i++ {
                r.Set(i).Next()
        }

        fmt.Println(r.SlicePrev())

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