-

```
package main

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

func main() {
        //
        s := gset.New(true)

        //
        s.Add(1)

        //
        s.Add([]interface{}{1, 2, 3}...)

        //
        fmt.Println(s.Size())

        //
        fmt.Println(s.Contains(2))

        // slice
        fmt.Println(s.Slice())

        //
        s.Remove(3)

        //
        s.Iterator(func(v interface{}) bool {
                fmt.Println("Iterator:", v)
                return true
        })

        //
        fmt.Println(s.String())

        //
        s.LockFunc(func(m map[interface{}]struct{}) {
                m[4] = struct{}{}
        })

        //
        s.RLockFunc(func(m map[interface{}]struct{}) {
                fmt.Println(m)
        })

        //
        s.Clear()
        fmt.Println(s.Size())
}
```

Content Menu

```
3
true
[1 2 3]
Iterator: 1
Iterator: 2
[1 2]
map[1:{} 2:{} 4:{}]
0
```

```
func (set *Set) Intersect(others ...*Set) (newSet *Set)
func (set *Set) Diff(others ...*Set) (newSet *Set)
func (set *Set) Union(others ...*Set) (newSet *Set)
func (set *Set) Complement(full *Set) (newSet *Set)
```

1. `Intersect`: setothers
2. `Diff`: setothers
3. `Union`: setothers
4. `Complement`: (: setfull)fullsetfullsetfullset.

```
package main

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

func main() {
        s1 := gset.NewFrom(g.Slice{1, 2, 3})
        s2 := gset.NewFrom(g.Slice{4, 5, 6})
        s3 := gset.NewFrom(g.Slice{1, 2, 3, 4, 5, 6, 7})

        //
        fmt.Println(s3.Intersect(s1).Slice())
        //
        fmt.Println(s3.Diff(s1).Slice())
        //
        fmt.Println(s1.Union(s2).Slice())
        //
        fmt.Println(s1.Complement(s3).Slice())
}
```

```
[1 2 3]
[4 5 6 7]
[1 2 3 4 5 6]
[7 4 5 6]
```

Contains/ContainsI

```
package main

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

func main() {
        var set gset.StrSet
        set.Add("a")
        fmt.Println(set.Contains("a"))
        fmt.Println(set.Contains("A"))
        fmt.Println(set.ContainsI("A"))

        // Output:
        // true
        // false
        // true
}
```

Pop/Pops

```
package main

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

func main() {
        var set gset.Set
        set.Add(1, 2, 3, 4)
        fmt.Println(set.Pop())
        fmt.Println(set.Pops(2))
        fmt.Println(set.Size())

        // May Output:
        // 1
        // [2 3]
        // 1
}
```

Join

```
package main

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

func main() {
        var set gset.Set
        set.Add("a", "b", "c", "d")
        fmt.Println(set.Join(","))

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

IsSubsetOf

```
package main

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

func main() {
        var s1, s2 gset.Set
        s1.Add(g.Slice{1, 2, 3}...)
        s2.Add(g.Slice{2, 3}...)
        fmt.Println(s1.IsSubsetOf(&s2))
        fmt.Println(s2.IsSubsetOf(&s1))

        // Output:
        // false
        // true
}
```

AddIfNotExist*

truefalse

- AddIfNotExist
- AddIfNotExistFunc
- AddIfNotExistFuncLock

```
package main

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

func main() {
        var set gset.Set
        fmt.Println(set.AddIfNotExist(1))
        fmt.Println(set.AddIfNotExist(1))
        fmt.Println(set.Slice())

        // Output:
        // true
        // false
        // [1]
}
```

Walk

```
package main

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

func main() {
        var (
                set    gset.StrSet
                names  = g.SliceStr{"user", "user_detail"}
                prefix = "gf_"
        )
        set.Add(names...)
        // Add prefix for given table names.
        set.Walk(func(item string) string {
                return prefix + item
        })
        fmt.Println(set.Slice())

        // May Output:
        // [gf_user gf_user_detail]
}
```

JSON/

gsetjson/

1. Marshal

```
package main

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

func main() {
        type Student struct {
                Id     int
                Name   string
                Scores *gset.IntSet
        }
        s := Student{
                Id:     1,
                Name:   "john",
                Scores: gset.NewIntSetFrom([]int{100, 99, 98}),
        }
        b, _ := json.Marshal(s)
        fmt.Println(string(b))
}
```

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

2. Unmarshal

```
package main


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


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

```
{1 john [100,99,98]}
```