-

https://pkg.go.dev/github.com/gogf/gf/v2/os/gtime

Content Menu

- New
- Now
- Format
- String
- Timestamp
- ToZone
- SetTimeZone
- StrToTime
- Add
- StartOfDay
- EndOfDay
- Month
- Second
- IsZero
- AddDate
- Equal
- Before
- After
- Layout
- IsLeapYear
- Date
- Datetime
- ISO8601
- RFC822
- StrToTimeFormat
- StrToTimeLayout
- MarshalJSON
- UnmarshalJSON
- WeekOfYear

New

- New Time
- :

```
func New(param ...interface{}) *Time
```

-

```
func ExampleNew() {
        t1 := gtime.New(time.Now())
        t2 := gtime.New("2018-08-08 08:08:08")
        t3 := gtime.New(1533686888)

        fmt.Println(t1)
        fmt.Println(t2)
        fmt.Println(t3)

        // Output:
        // 2021-11-18 14:18:27
        // 2018-08-08 08:08:08
        // 2018-08-08 08:08:08
```

Now

- : Now
-

```
func Now() *Time
```

-

```
func ExampleNow() {
        t := gtime.Now()
        fmt.Println(t)

        // Output:
        // 2021-11-06 13:41:08
}
```

Format

- :
-

```
func (t *Time) Format(format string) string
```

- -

```
func ExampleTime_Format() {
        gt1 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.Format("Y-m-d"))
        fmt.Println(gt1.Format("l"))
        fmt.Println(gt1.Format("F j, Y, g:i a"))
        fmt.Println(gt1.Format("j, n, Y"))
        fmt.Println(gt1.Format("h-i-s, j-m-y, it is w Day z"))
        fmt.Println(gt1.Format("D M j G:i:s T Y"))

        // Output:
        // 2018-08-08
        // Wednesday
        // August 8, 2018, 8:08 am
        // 8, 8, 2018
        // 08-08-08, 8-08-18, 0831 0808 3 Wedam18 219
        // Wed Aug 8 8:08:08 CST 2018
}
```

String

- :
-

```
func (t *Time) String() string
```

-

```
func ExampleTime_String() {
        gt := gtime.New("2018-08-08 08:08:08")
        t1 := gt.String()

        fmt.Println(t1)
        fmt.Println(reflect.TypeOf(t1))

        // Output:
        // 2018-08-08 08:08:08
        // string
}
```

Timestamp

- : TimestampMicro/TimestampMilli/TimestampNano
-

```
func (t *Time) Timestamp() int64
func Timestamp() int64
```

-

```
func ExampleTime_Timestamp() {
        t := gtime.Now()

        fmt.Println(t.Timestamp())
    fmt.Println(gtime.Timestamp())
    fmt.Println(t.TimestampMicro())
         fmt.Println(t.TimestampMilli())
        fmt.Println(t.TimestampNano())

        // Output:
        // 1533686888
    // 1533686888
    // 1533686888000
        // 1533686888000000
        // 1533686888000000000
}
```

ToZone

- :
-

```
func (t *Time) ToZone(zone string) (*Time, error)
```

-

```
func ExampleTime_ToZone() {
        gt1 := gtime.Now()
        gt2, _ := gt1.ToZone("Asia/Shanghai")
        gt3, _ := gt1.ToZone("Asia/Tokyo")

        fmt.Println(gt2)
        fmt.Println(gt3)

        // May Output:
        // 2021-11-11 17:10:10
        // 2021-11-11 18:10:10
}
```

SetTimeZone

- :
-

```
func SetTimeZone(zone string) error
```

-

```
func ExampleSetTimeZone() {
        gtime.SetTimeZone("Asia/Shanghai")
        fmt.Println(gtime.Datetime())

        gtime.SetTimeZone("Asia/Tokyo")
        fmt.Println(gtime.Datetime())
        // May Output:
        // 2018-08-08 08:08:08
        // 2018-08-08 09:08:08
}
```

StrToTime

- :
-

```
func StrToTime(str string, format ...string) (*Time, error)
```

-

```
func ExampleStrToTime() {
        res, _ := gtime.StrToTime("2006-01-02T15:04:05-07:00", "Y-m-
d H:i:s")
        fmt.Println(res)

        // May Output:
        // 2006-01-02 15:04:05
}
```

Add

- :
-

```
func (t *Time) Add(d time.Duration) *Time
```

-

```
func ExampleTime_Add() {
        gt := gtime.New("2018-08-08 08:08:08")
        gt1 := gt.Add(time.Duration(10) * time.Second)

        fmt.Println(gt1)

        // Output:
        // 2018-08-08 08:08:18
}
```

StartOfDay

- : StartOfHalf/StartOfHour/StartOfMonth/StartOfMinute/StartOfQuarter
-

```
func (t *Time) StartOfDay() *Time
```

-

```
func ExampleTime_StartOfDay() {
        gt1 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.StartOfDay())

        // Output:
        // 2018-08-08 00:00:00
}
```

EndOfDay

- : EndOfHalf/EndOfHour/EndOfMonth/EndOfMinute/EndOfQuarter

-

```
func (t *Time) EndOfDay() *Time
```

-

```
func ExampleTime_EndOfDay() {
        gt1 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.EndOfDay())

        // Output:
        // 2018-08-08 23:59:59
}
```

Month

- : 11
-

```
func (t *Time) Month() int
```

-

```
func ExampleTime_Month() {
        gt := gtime.New("2018-08-08 08:08:08")
        t1 := gt.Month()

        fmt.Println(t1)

        // Output:
        // 8
}
```

Second

- : 10:10:08 8
-

```
func (t *Time) Second() int
```

-

```
func ExampleTime_Second() {
        gt := gtime.New("2018-08-08 08:08:08")
        t1 := gt.Second()

        fmt.Println(t1)

        // Output:
        // 8
}
```

IsZero

- : 0001-01-01 00:00:00O, O1970-01-01 08:00:00
-

```
func (t *Time) IsZero() bool
```

-

```
func ExampleTime_IsZero() {
        gt := gtime.New("0-0-0")

        fmt.Println(gt.IsZero())

        // Output:
        // true
}
```

AddDate

- :
-

```
func (t *Time) AddDate(years int, months int, days int) *Time
```

-

```
func ExampleTime_AddDate() {
        var (
                year  = 1
                month = 2
                day   = 3
        )
        gt := gtime.New("2018-08-08 08:08:08")
        gt = gt.AddDate(year, month, day)

        fmt.Println(gt)

        // Output:
        // 2019-10-11 08:08:08
}
```

Equal

- :
-

```
func (t *Time) Equal(u *Time) bool
```

-

```
func ExampleTime_Equal() {
        gt1 := gtime.New("2018-08-08 08:08:08")
        gt2 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.Equal(gt2))

        // Output:
        // true
}
```

Before

- :
-

```
func (t *Time) Before(u *Time) bool
```

-

```
func ExampleTime_Before() {
        gt1 := gtime.New("2018-08-07 08:08:08")
        gt2 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.Before(gt2))

        // Output:
        // true
}
```

After

- :
-

```
func (t *Time) After(u *Time) bool
```

-

```
func ExampleTime_After() {
        gt1 := gtime.New("2018-08-07 08:08:08")
        gt2 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.After(gt2))

        // Output:
        // false
}
```

Layout

- :
-

```
func (t *Time) Layout(layout string) string
```

-

```
func ExampleTime_Layout() {
        gt1 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.Layout("2006-01-02"))

        // Output:
        // 2018-08-08
}
```

IsLeapYear

- :
-

```
func (t *Time) IsLeapYear() bool
```

-

```
func ExampleTime_IsLeapYear() {
        gt1 := gtime.New("2018-08-08 08:08:08")

        fmt.Println(gt1.IsLeapYear())

        // Output:
        // false
}
```

Date

- :
-

```
func Date() string
```

-

```
func ExampleDate() {
        fmt.Println(gtime.Date())

        // May Output:
        // 2006-01-02
}
```

Datetime

- :
-

```
func Datetime() string
```

-

```
func ExampleDatetime() {
        fmt.Println(gtime.Datetime())

        // May Output:
        // 2006-01-02 15:04:05
}
```

ISO8601

- : ISO8601
-

```
func ISO8601() string
```

-

```
func ExampleISO8601() {
        fmt.Println(gtime.ISO8601())

        // May Output:
        // 2006-01-02T15:04:05-07:00
}
```

RFC822

- : RFC822
-

```
func RFC822() string
```

-

```
func ExampleRFC822() {
        fmt.Println(gtime.RFC822())

        // May Output:
        // Mon, 02 Jan 06 15:04 MST
}
```

StrToTimeFormat

- : StrToTimeFormat
-

```
func StrToTimeFormat(str string, format string) (*Time, error)
```

-

```
func ExampleStrToTimeFormat() {
        res, _ := gtime.StrToTimeFormat("2006-01-02 15:04:05", "Y-m-
d H:i:s")
        fmt.Println(res)

        // Output:
        // 2006-01-02 15:04:05
}
```

StrToTimeLayout

- : StrToTimeLayout
-

```
func StrToTimeLayout(str string, layout string) (*Time, error)
```

-

```
func ExampleStrToTimeLayout() {
        res, _ := gtime.StrToTimeLayout("2018-08-08", "2006-01-02")
        fmt.Println(res)

        // Output:
        // 2018-08-08 00:00:00
}
```

MarshalJSON

- : MarshalJSONjson.Marshal
-

```
func (t *Time) MarshalJSON() ([]byte, error)
```

-

```
func ExampleTime_MarshalJSON() {
        type Person struct {
                Name string                `json:"name"`
                Birthday *gtime.Time        `json:"birthday"`
        }
        p := new(Person)
        p.Name = "goframe"
        p.Birthday = gtime.New("2018-08-08 08:08:08")
        j, _ := json.Marshal(p)
        fmt.Println(string(j))

        // Output:
        // {"name":"xiaoming","birthday":"2018-08-08 08:08:08"}
}
```

UnmarshalJSON

- : UnmarshalJSONjson.Unmarshal
-

```
func (t *Time) UnmarshalJSON() ([]byte, error)
```

-

```
func ExampleTime_MarshalJSON() {
        type Person struct {
                Name string                `json:"name"`
                Birthday *gtime.Time        `json:"birthday"`
        }
        p := new(Person)
        p.Name = "goframe"
        p.Birthday = gtime.New("2018-08-08 08:08:08")
        j, _ := json.Marshal(p)
        fmt.Println(string(j))

        // Output:
        // {"name":"xiaoming","birthday":"2018-08-08 08:08:08"}
}
```

WeekOfYear

- : WeekOfYear1DayOfYear/DaysInMonth
-

```
func (t *Time) WeeksOfYear() int
```

-

```
func ExampleTime_WeeksOfYear() {
        gt1 := gtime.New("2018-01-08 08:08:08")

        fmt.Println(gt1.WeeksOfYear())

        // Output:
        // 2
}D
```