Go语言 标准库 Time包

time包提供了时间的显示和测量用的函数。日历的计算采用的是公历。

时间类型

time.Time类型表示时间。我们可以通过time.Now()函数获取当前的时间对象，然后获取时间对象的年月日时分秒等信息。

```
func timeDemo() {
    now := time.Now() //获取当前时间
    fmt.Printf("当前时间:%v\n", now)

    year := now.Year()     //年
    month := now.Month()   //月
    day := now.Day()       //日
    hour := now.Hour()     //小时
    minute := now.Minute() //分钟
    second := now.Second() //秒
    fmt.Printf("%d-%02d-%02d %02d:%02d:%02d\n", year, month, day, hour, minute,
second)
    fmt.Printf("%T,%T,%T,%T,%T,%T,%T\n", now, year, month, day, hour, minute,
second)
}
```

时间戳

时间戳是自1970年1月1日（08:00:00GMT）至当前时间的总毫秒数。它也被称为Unix时间戳（UnixTimestamp）。基于时间对象获取时间戳的示例代码如下：

```
func timestampDemo() {
    now := time.Now()            //获取当前时间
    timestamp1 := now.Unix()     //时间戳
    timestamp2 := now.UnixNano() //纳秒时间戳
    fmt.Printf("current timestamp1:%v\n", timestamp1)
    fmt.Printf("current timestamp2:%v\n", timestamp2)
}
```

使用`time.Unix()`函数可以将时间戳转为时间格式。

```
func timestampToDateDemo(timestamp int64) {
    timeObj := time.Unix(timestamp, 0) //将时间戳转为时间格式
    fmt.Println(timeObj)
    year := timeObj.Year()     //年
    month := timeObj.Month()   //月
    day := timeObj.Day()       //日
    hour := timeObj.Hour()     //小时
    minute := timeObj.Minute() //分钟
    second := timeObj.Second() //秒
    fmt.Printf("%d-%02d-%02d %02d:%02d:%02d\n", year, month, day, hour, minute,
second)
}
```

时间间隔

time.Duration是time包定义的一个类型，它代表两个时间点之间经过的时间，以纳秒为单位。time.Duration表示一段时间间隔，可表示的最长时间段大约290年。

time包中定义的时间间隔类型的常量如下：

```
const (
    Nanosecond  Duration = 1
    Microsecond          = 1000 * Nanosecond
    Millisecond          = 1000 * Microsecond
    Second               = 1000 * Millisecond
    Minute               = 60 * Second
    Hour                 = 60 * Minute
)
```

例如：time.Duration表示1纳秒，time.Second表示1秒。

时间操作

Add

我们在日常的编码过程中可能会遇到要求时间+时间间隔的需求，Go语言的时间对象有提供Add方法如下：

```
func (t Time) Add(d Duration) Time
```

求一个小时之后的时间：

```
func timeAddDemo() {
    now := time.Now()
    later := now.Add(time.Hour) // 当前时间加1小时后的时间
    fmt.Println(later)
}
```

Sub

求两个时间之间的差值：

```
func (t Time) Sub(u Time) Duration
```

返回一个时间段t-u。如果结果超出了Duration可以表示的最大值/最小值，将返回最大值/最小值。要获取时间点t-d（d为Duration），可以使用t.Add(-d)。

```
func timeSubDemo() {
    now := time.Now()
    fmt.Println(now)

    targetTime := now.Add(time.Hour)
    // 目标时间与此时相比相差1h0m0s
    fmt.Println(targetTime.Sub(now)) //1h0m0s

    before := now.Add(time.Hour * -1) // 当前时间减1小时后的时间
    fmt.Println(before)
}
```

Equal

判断两个时间是否相同，会考虑时区的影响，因此不同时区标准的时间也可以正确比较。本方法和用t==u不同，这种方法还会比较地点和时区信息。

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

Before

如果t代表的时间点在u之前，返回真；否则返回假。

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

After

如果t代表的时间点在u之后，返回真；否则返回假。

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

定时器

使用time.Tick(时间间隔)来设置定时器，定时器的本质上是一个通道（channel）。

```
func tickDemo() {
    ticker := time.Tick(time.Second) //定义一个1秒间隔的定时器
    for i := range ticker {
        fmt.Println(i)//每秒都会执行的任务
    }
}
```

时间格式化

时间类型有一个自带的方法Format进行格式化，需要注意的是Go语言中格式化时间模板不是常见的Y-m-d H:M:S而是使用Go的诞生时间2006年1月2号15点04分（记忆口诀为2006 1 2 3 4）。ps：真不要脸！

如果想格式化为12小时方式，需指定PM。

```
func formatDemo() {
    now := time.Now()
    // 格式化的模板为Go的出生时间2006年1月2号15点04分 Mon Jan
    // 24小时制
    fmt.Println(now.Format("2006-01-02 15:04:05.000 Mon Jan"))
    // 12小时制
    fmt.Println(now.Format("2006-01-02 03:04:05.000 PM Mon Jan"))
    fmt.Println(now.Format("2006/01/02 15:04"))
    fmt.Println(now.Format("15:04 2006/01/02"))
    fmt.Println(now.Format("2006/01/02"))
}
```

解析字符串格式的时间

```
func strToDateDemo() {
    now := time.Now()
    fmt.Println(now)
    // 加载时区
    loc, err := time.LoadLocation("Asia/Shanghai")
    if err != nil {
        fmt.Println(err)
        return
    }
    // 按照指定时区和指定格式解析字符串时间
    timeObj, err := time.ParseInLocation("2006/01/02 15:04:05", "2022/12/04
14:15:20", loc)
    if err != nil {
        fmt.Println(err)
        return
    }
    fmt.Println(timeObj)
    fmt.Println(timeObj.Sub(now))
}
```