Golang time 包以及日期函数

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

一、time 包

时间和日期是我们编程中经常会用到的，在 golang 中 time 包提供了时间的显示和测量用的函数。

二、time.Now()获取当前时间

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。示例代码如下：

```
func main() {
    now := time.Now() //获取当前时间
    fmt.Printf("current time:%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)
}
```

注意：%02d 中的 2 表示宽度，如果整数不够 2 列就补上 0

三、Format 方法格式化输出日期字符串

```
func main() {

    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"))

    // 12 小时制

    fmt.Println(now.Format("2006-01-02 03:04:05"))

    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"))

}
```

四、获取当前的时间戳

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：

```
package main
import (
    "fmt"
    "time"
)
func main() {
    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)
}
```

五、时间戳转换为日期字符串（年-月-日 时:分:秒）

使用 time.Unix()函数可以将时间戳转为时间格式。

```
package main
import (
    "fmt"
    "time"
)
func unixToTime(timestamp int64) {
    timeObj := time.Unix(timestamp, 0) //将时间戳转为时间格式
    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, secon
d)
}
func main() {
    unixToTime(1587880013)
}
```

六、now.Format 把时间戳格式化成日期

```
var timestamp int64 = 1587880013    //时间戳
t := time.Unix(timestamp, 0)          //日期对象
fmt.Println(t.Format("2006-01-02 03:04:05"))    //日期格式化输出
```

七、日期字符串转换成时间戳

```
func main() {
    t1 := "2019-01-08 13:50:30"                                    间字符串
    timeTemplate := "2006-01-02 15:04:05"                          //常规类型
    stamp, _ := time.ParseInLocation(timeTemplate , t1, time.Local)
    fmt.Println(stamp.Unix())
}
 //输出：1546926630
```

八、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。

八、时间操作函数

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
```

举个例子，求一个小时之后的时间：

```
func main() {
    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)。

Equal

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

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。

Before

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

如果 t 代表的时间点在 u 之前，返回真；否则返回假。

After

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

如果 t 代表的时间点在 u 之后，返回真；否则返回假。

九、定时器

1、使用 time.NewTicker(时间间隔)来设置定时器

```
ticker := time.NewTicker(time.Second) //定义一个 1 秒间隔的定时器
n := 0
for i := range ticker.C {
    fmt.Println(i) //每秒都会执行的任务
    n++
    if n > 5 {
        ticker.Stop()
        return
    }
}
```

2、time.Sleep(time.Second) 来实现定时器

```
for {
    time.Sleep(time.Second)
    fmt.Println("我在定时执行任务")
}
```

十、练习题

1、获取当前时间，格式化输出为 2020/06/19 20:30:05 格式。
2、获取当前时间，格式化输出为时间戳
3、把时间戳 1587880013 转换成日期字符串，格式为 2020/xx/xx xx:xx:xx
4、把日期字符串 2020/06/19 20:30:05 转换成时间戳
5、编写程序统计一段代码的执行耗时时间，单位精确到微秒。