05

Formatando tempo em Go

Transcrição

Queremos imprimir a data no log no seguinte formato:

```
05/06/2017 10:50:06 - https://www.alura.com.br - online: true
```

Para trabalhar com tempo em Go, vamos utilizar o já conhecido pacote `time`. Para pegar o tempo atual, ele possui a função `Now`, que nos retorna um objeto, portanto vamos formatá-lo, com a função `format`.

Para formatar a data, essa função utiliza diversas constantes, que podem ser consultadas no seu código fonte (https://golang.org/src/time/format.go):

```
const (
        _                       = iota
        stdLongMonth            = iota + stdNeedDate  // "January"
        stdMonth                                      // "Jan"
        stdNumMonth                                   // "1"
        stdZeroMonth                                  // "01"
        stdLongWeekDay                                // "Monday"
        stdWeekDay                                    // "Mon"
        stdDay                                        // "2"
        stdUnderDay                                   // "_2"
        stdZeroDay                                    // "02"
        stdHour                 = iota + stdNeedClock // "15"
        stdHour12                                     // "3"
        stdZeroHour12                                 // "03"
        stdMinute                                     // "4"
        stdZeroMinute                                 // "04"
        stdSecond                                     // "5"
        stdZeroSecond                                 // "05"
        stdLongYear             = iota + stdNeedDate  // "2006"
        stdYear                                       // "06"
        stdPM                   = iota + stdNeedClock // "PM"
        stdpm                                         // "pm"
        stdTZ                   = iota                // "MST"
        stdISO8601TZ                                  // "Z0700"  // prints Z for UTC
        stdISO8601SecondsTZ                           // "Z070000"
        stdISO8601ShortTZ                             // "Z07"
        stdISO8601ColonTZ                             // "Z07:00" // prints Z for UTC
        stdISO8601ColonSecondsTZ                      // "Z07:00:00"
        stdNumTZ                                      // "-0700"  // always numeric
        stdNumSecondsTz                               // "-070000"
        stdNumShortTZ                                 // "-07"    // always numeric
        stdNumColonTZ                                 // "-07:00" // always numeric
        stdNumColonSecondsTZ                          // "-07:00:00"
        stdFracSecond0                                // ".0", ".00", ... , trailing zeros included
        stdFracSecond9                                // ".9", ".99", ..., trailing zeros omitted

        stdNeedDate  = 1 << 8             // need month, day, year
        stdNeedClock = 2 << 8             // need hour, minute, second
```

```
        stdArgShift  = 16                // extra argument in high bits, above low stdArgShift
        stdMask      = 1<<stdArgShift - 1 // mask out argument
    )
```

Como queremos representar dia, mês e ano no formato numérico, vamos utilizar o `stdZeroDay`, representado pelo `02`, `stdZeroMonth`, representado pelo `01` e `stdLongYear`, representado pelo `2006`:

```
// restante do código omitido

func registraLog(site string, status bool) {

    arquivo, err := os.OpenFile("log.txt", os.O_CREATE|os.O_RDWR|os.O_APPEND, 0666)

    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    arquivo.WriteString(time.Now().Format("02/01/2006") + " - " + site + " - online: " +
        strconv.FormatBool(status) + "\n")

    arquivo.Close()
}
```

Falta ajustar a hora, vamos utilizar o `stdHour`, representado pelo `15`, `stdZeroMinute`, representado pelo `04`, e `stdZeroSecond`, representado pelo `05`:

```
// restante do código omitido

func registraLog(site string, status bool) {

    arquivo, err := os.OpenFile("log.txt", os.O_CREATE|os.O_RDWR|os.O_APPEND, 0666)


    if err != nil {
        fmt.Println("Ocorreu um erro:", err)
    }

    arquivo.WriteString(time.Now().Format("02/01/2006 15:04:05") + " - " + site +
        " - online: " + strconv.FormatBool(status) + "\n")

    arquivo.Close()
}
```

Podemos agora executar o monitoramento e verificar o log, ele está exatamente como queríamos! Agora que temos a funcionalidade de salvar os logs pronta, falta exibi-los ao escolher o comando `2`, e é justamente isso que faremos no próximo vídeo.