MAVENIR™
Trust the Future

Golang Training Date and Time

24/01/2023

Golang: Date and Time

Overview

- Panic in golang is similar to the exception
- Panic is meant to exit from a program in abnormal conditions.
- Panic can occur in a program in two ways
  - Runtime error in the program
  - By calling the panic function explicitly

```
func panic(v interface{})
```

Golang: Date and Time

Overview

- **Location value in time.Time struct is used evaluate Minute, Hour, Month and Day**

Golang: Date and Time

Creating Time

> func Now() Time
  - This function is used to get the current local time stamp

> func Date(year int, month Month, day, hour, min, sec, nsec int, loc *Location) Time
  - This function returns Time yyyy-mm-dd hh:mm:ss + nsec nanoseconds
  - Location paramaeter is used for timezone
  - Nil location stands for UTC time

Golang: Date and Time

Duration

- Time elapsed between 2 instants of time
- Represented as int64 nanosecond count

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

MAVENIR

Golang: Date and Time

Duration

- Functions returning duration
  - **func (t Time) Sub(u Time) Duration //**It returns the duration t-u
  - **func Since(t Time) Duration //** It returns the duration which has elapsed since t
  - **func Until(t Time) Duration //**It returns the duration until t

Golang: Date and Time

Working with time

- Add () function is used to add/subtract duration to a time.
  - func (t Time) Add(d Duration) Time
- AddDate() function is used to add/subtract years, months and days to time t.
  - func (t Time) AddDate(years int, months int, days int) Time

Golang: Date and Time

Parsing Time

- Golang uses codes for date and time place holder

Mon Jan 2 15:04:05 MST 2006 (MST is GMT-0700)

or

01/02 03:04:05PM '06 -0700

- func Parse(layout, value string) (Time, error) is used for parasing time
  - Layout represents time format and value represents time in string

Golang: Date and time

Parsing

> For parsing **2023-02-16**, layout string should be **06-01-02** or **2006-01-02** or something which maps correctly based on above placeholder table.

> Similarly for parsing **“2023-Feb-16 Wednesday 12:19:25”** the layout string can be **“2006-Jan-02 Monday 03:04:05”**

MAVENIR

Golang: Date and Time

Date and Time Formatting

- func (t Time) Format(layout string) is used to represent time in string format specified by layout
- It is performs reverse action of parse function
- Layout for mats are similar to parse layout format

Golang: Date and Time

Time Conversion

- To support legacy systems and protocol requirements, time conversion to UNIX time become essential
- time.Time to Unix Timestamp
- Unix Timestamp to time.Time

Golang: Date and Time

Time zone

- The **In** function is used to change the **location** associated with a particular **time.Time** object.
- In function returns
  - A copy of **t** is created representing the same time instant.
  - t's location is set to the location passed to In function for display purposes
  - **t** is returned back
- time.LoadLocation() is used to create location
- Time zone strings are form TZ Database
  - https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

MAVENIR

Golang: Timer and Ticker

Overview

- Timers in language and systems are essentials
  - Timer is for onetime expiry
  - Ticket is for repeating timer after interval
- Timer represents a single event in the future
- time.NewTimer() function is used to create timer
  - Duration is parameter and returns timer object
  - Timer.c is channel

Golang: Timer and Ticker

Overview

- Tickers are similar to timer with a difference that ticker is repeating timer
- Time.NewTicker is used to create ticker

Golang: Timer and Ticker

Overview

- Time.After ()
- time.Tick(time.Duration(interval) * time.Second)
- time.AfterFunc(3*time.Second, func()