MAVENIR™
Trust the Future

Golang Training Pointers

24/01/2023

Golang: Error Handling..

Panic and Recover

- 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: Error Handling

Panic and Recover

- **Runtime Error Panic**
  - Out of bounds array access
  - Calling a function on a nil pointer
  - Sending on a closed channel
  - Incorrect type assertion
- **Calling the panic function explicitly**
  - The function expected a valid argument but instead, a nil argument was supplied
  - Any other scenario in which the program cannot continue

MAVENIR

Golang: Error Handling

Panic with defer

- When panic is raised in a function, execution of that function is stops
- Deferred function will be executed
- Deferred function of all the function calls in the stack will also be executed until all the functions have returned

Golang: Error Handling

Recover in golang

- Go provides a built-in function **recover** for recovering from a panic

  func recover() interface{}

- Recover should be handled in defer function

Golang: Error

Overview

- Error handling can be done
  - Using type which implements **error** interface, it is a conventional way to represent an error
  - Using panic and recover

```
type error interface {
    Error() string
}
```

Golang: Error

Advantages of Error handling

- It allows more control over handling the error. The error can be checked at each step
- Prevents the ugly code of try-catch and exception handling
- Ways of creating an error
  - **errors.New(“some_error_message”)**
  - **fmt.Errorf(“error is %s”, “some_error_message”)**
  - **Creating Custom error**

Golang: Interfaces

Adding Items

> Polymorphism

Golang: Goroutine

Overview

- A Goroutine is a function or method which executes independently
- Goroutine can be compared as a light weighted thread
- Every program contains at least a single Goroutine and that Goroutine is known as the **main Goroutine**
- All the Goroutines are working under the main Goroutines
- if the main Goroutine terminated, then all the goroutine present in the program also terminated

Golang: Goroutine

Delete

Syntax

```
func name(){

// statements

}


// using go keyword as the

// prefix of your function call

go name()
```

Golang: Goroutine

Anonymous Goroutine

- Goroutine with an anonymous function can be created
- Anonymous Goroutine simply by using go keyword as a prefix

Golang: Goroutine

Sync package

- Goroutine can synchronize use sync.Mutex //Smillar to posix mutex
- Goroutines can wait for each other using Wait Groups
- Critical sections using mutexes

Golang: Channels

Overview

- Channel are used to share data between goroutines
- Channels act as a pipe between the goroutines
- Channels guarantees a synchronous exchange.
- Data type should be specified at the time of declaration of a channel
- Values and pointers of built-in, named, struct, and reference types can be shared across channel
- Only one goroutine has access to a data item at any given time
- Hence data races cannot occur, by design

Golang: Channel

Types

- Channels are of 2 types
  - Unbuffered channel
    - Unbuffered channel will have only one item, this is used for synchronous communication
  - Buffered channels
    - Buffer channels will have specified number of channles, and are used for Asynchronous communication

Golang: Log

Overview

- The standard library package log provides a basic infrastructure for log management.
- Logs can be providing code tracing, profiling, and analytics
- log.SetPrefix("LOG: ")
- log.SetFlags(log.Ldate | log.Lmicroseconds | log.Llongfile)
- log.Println("init started")
- log.Fatalln("fatal message")
- log.Panicln("panic message")

Golang: Files and Directories

Overview

- OS Package provides functionality to work with files
  - Os.Create
  - _, err := os.Stat("test")
  - os.IsNotExist(err)
  - os.Rename(oldName, newName)
  - Os.Open
  - Io.Copy
  - os.Remove

Golang: Files and Directories

File stat

- fileStat.Name()
- fileStat.Size()
- fileStat.Mode()
- fileStat.ModTime()
- fileStat.IsDir()
- Os.Truncate
- File permission (os.O_RDWR|os.O_APPEND|os.O_CREATE)
- Os.Chmod