Golang Session

- Harsh Dusane

Topic : Panic and Recover

Panic

- The built-in type system of GO Language catches many mistakes at compile time, but unable to check mistakes like an out-of-bounds array, access or nil pointer deference which require checks at run time. GO does not have an exception mechanism you can't throw exceptions. During the execution when Go detects these mistakes, it panics and stops all normal execution, all deferred function calls in that goroutine are executed and finally program crashes with a log message. This log message usually has enough information to analyze the root cause of the problem without running the program repeatedly, so it should always be included in a bug report about a panicking program.

Panic

- Panic is a built-in function that stops the ordinary flow of control and begins panicking. When the function X calls panic, execution of X stops, any deferred functions in X are executed normally, and then X returns to its caller. To the caller, X then behaves like a call to panic. The process continues up the stack until all functions in the current goroutine have returned, at which point the program crashes. Panics can be initiated by invoking panic directly. They can also be caused by run-time errors, such as out-of-bounds array accesses.

Sample code

```
package main
import "fmt"
func main() {
        var action int
    fmt.Println("Enter 1 for Student and 2 for Professional")
    fmt.Scanln(&action)
    /*  Use of Switch Case in Golang */
    switch action {
        case 1:
            fmt.Printf("I am a  Student")
        case 2:
            fmt.Printf("I am a  Professional")
                default:
                        panic(fmt.Sprintf("I am a  %d",action))
    }
    fmt.Println("")
    fmt.Println("Enter 1 for US and 2 for UK")
    fmt.Scanln(&action)
    /*  Use of Switch Case in Golang */
    switch       {
        case 1:
            fmt.Printf("US")
        case 2:
            fmt.Printf("UK")
                default:
                        panic(fmt.Sprintf("I am a  %d",action))
    }
}
```

Recover

- Recover is a built-in function that regains control of a panicking goroutine. Recover is only useful inside deferred functions. During normal execution, a call to recover will return nil and have no other effect. If the current goroutine is panicking, a call to recover will capture the value given to panic and resume normal execution.

Sample Code

```
package main
import "fmt"
func main() {
        var action int
    fmt.Println("Enter 1 for Student and 2 for Professional")
    fmt.Scanln(&action)
    /*  Use of Switch Case in Golang */
    switch action {
        case 1:
            fmt.Printf("I am a  Student")
        case 2:
            fmt.Printf("I am a  Professional")
                default:
                        panic(fmt.Sprintf("I am a  %d",action))
    }
    defer func() {
        action := recover()
                fmt.Println(action)
    }()
}
```