



Backend Programming

M. Yauri M. Attamimi

Upgrade your potential !




Foundation Class

- WEEK 2 (Session 2) -



Package

- 
1. Package Declaration
 2. Init Function
 3. Importing Package
 4. Export and Import Types

Package Declaration

- Packages are used to organize source code for better readability and maintainability hence it can be easily reused.
- Every executable go application must contain a main function. This function is the entry point for execution. The main function should reside in the main package.
- The line of code to specify that a particular source file belongs to a particular package is:
`package <package_name>` which should be written on the first line of every go source file.



Init Function

- Every package contains an init function. The init function should not have any return type and should not have any parameters.
- The init function cannot be explicitly called in our source code.
- Here what it looks like :

```
func init() {  
}
```
- The init function can be used to perform initialization tasks and can also be used to verify the correctness of the program before the execution starts.
- The order of initialization of a package is as follows:
 1. Package level variables are initialized first
 2. init function is called next
- if a package imports other packages, the imported packages are initialized first.
- A package will be initialized only once even if it is imported from multiple packages.



Export and Import Types

- We can make any type to be publicly available (exported types) by capitalized their first letter. e.g.:

```
// exported function (can be accessed from outside the package / publicly available)  
func CountTotalOrder() int
```

```
// un-exported function (cannot be accessed from outside the package / private)  
func countTotalOrder() int
```

- Any variables or functions which starts with a capital letter are exported names in Go, hence it can be used (imported) from other packages (outside of the package where they were being defined).





Reading and Writing Files

Writing to File (I)

```
import (  
    "io/ioutil"  
    "fmt"  
)  
  
data := []byte("Init data")  
err := ioutil.WriteFile("localfile.data", data, 0777)  
if err != nil {  
    fmt.Println(err)  
}
```



Writing to File (II)

```
import (  
    "io/ioutil"  
    "os"  
)  
  
f, err := os.OpenFile(  
    "localfile.data", os.O_APPEND|os.O_WRONLY, 0600)  
if err != nil {  
    fmt.Println(err)  
}  
defer f.Close()  
if _, err = f.WriteString("\nnew line\n"); err != nil {  
    fmt.Println(err)  
}
```



Reading from File

```
import (  
    "io/ioutil"  
    "fmt"  
)  
  
data, err := ioutil.ReadFile("localfile.data")  
if err != nil {  
    fmt.Println(err)  
}  
  
fmt.Println(string(data))
```





Check the exercises at
<https://exercism.io>





SUMMARY

You have learned the concept of package in Go, why do we need it and how to use it properly in order to make your code more maintainable, including various concepts related to package. You have also learned about basic concepts of concurrency and how we handle concurrency things in Go.





Thank You