How To Read Multiple Times From An Io Reader In Golang

In this article we are going to look at how we can efficiently work with io.Reader(s). In particular we are going to see how you can read multiple times from an io.Reader. This can be for example when reading or partially reading the response body of a HTTP Request or simply the need to read a file or data from a network connection multiple times. With the highly composable nature of the Go standard library we can easily achieve this by using either of the following

Using io.TeeReader in Go (Golang)

The io.TeeReader package was inspired by the tee unix command). The tee unix command reads from standard input and writes to standard output and one or more files, effectively duplicating the input stream. This is what TeeReader does to certain extents, let’s have a look at the official Go doc

```
func TeeReader(r Reader, w Writer) Reader
```

> TeeReader returns a Reader that writes to w what it reads from r. All reads from r performed through it are matched with corresponding writes to w. There is no internal buffering - the write must complete before the read completes. Any error encountered while writing is reported as a read error.

Example of how the io.TeeReader works

```
package main

import (
    "fmt"
    "io/ioutil"
    "io"
    "bytes"
    "strings"
)

func main() {
    reader := strings.NewReader("the quick brown fox jumps over the lazy dog")
    buf := &bytes.Buffer{}
    tee := io.TeeReader(reader, buf)
    reader1, _ := ioutil.ReadAll(tee)
    reader2, _ := ioutil.ReadAll(buf)
    fmt.Println(string(reader1))
    fmt.Println(string(reader2))
}
```

Expected Output

```
the quick brown fox jumps over the lazy dog
the quick brown fox jumps over the lazy dog
```

You can also try it here on the Go playground https://play.golang.org/p/XtUMQtMdXjn

NB When using the io.TeeReader remember to read ALL data from the input reader as the data is copied as long as is read. If you only read a portion of the input stream the TeeReader is going to copy only that portion into the buffer!

Using io.ReadSeeker in Go (Golang)

The io.ReadSeeker in Go is a combination of the io.Reader interface and the io.Seekerinterface. This allows us to re-wind a reader to a specific location of the stream by passing an offset relative to the start, current location or end of the stream (which can be a file stream, a string, a network connection or anything that implements an io.Reader). Let's see first the signature of the io.Seeker Seek method

```
type Seeker interface {
    Seek(offset int64, whence int) (int64, error)
}
```

> Seek sets the offset for the next Read or Write to offset, interpreted according to whence: SeekStart means relative to the start of the file, SeekCurrent means relative to the current offset, and SeekEnd means relative to the end. Seek returns the new offset relative to the start of the file and an error, if any.

Let’s see an example on how we can read multiple times from the same io.Reader using the io.Seeker interface

```
package main

import (
    "fmt"
    "io/ioutil"
    "io"
    "strings"
)

func main() {
    reader := strings.NewReader("the quick brown fox jumps over the lazy dog")
    reader1, _ := ioutil.ReadAll(reader)
    reader.Seek(0, io.SeekStart)
    reader2, _ := ioutil.ReadAll(reader)
    fmt.Println(string(reader1))
    fmt.Println(string(reader2))
}
```

You can also try it here on the Go playground https://play.golang.org/p/pLPbo5dwaFt

NB: the input stream or reader you want to read from and re-wind must implement the io.Seeker interface, for example the os.File and strings.Reader do implement the io.Seeker interface, but you might want to check your specific case