Go语言 标准库 ioutil包

ioutil包封装了一些I/O实用程序函数。

函数	说明
ReadAll	读取数据，返回读到的字节slice
ReadDir	读取一个目录，返回目录入口数组[]os.FileInfo
ReadFile	都一个文件，返回读到的文件内容字节slice
WriteFile	根据文件路径，写入字节slice
TempDir	在一个目录中创建指定前缀名的临时目录，返回新临时目录的路径
TempFile	在一个目录中创建指定前缀名的临时文件，返回os.File
NopCloser	用一个无操作的Close方法包装文件并返回一个ReadCloser接口。

NopCloser

```
func NopCloser(r io.Reader) io.ReadCloser
```

用一个无操作的Close方法包装r返回一个ReadCloser接口。

示例：

```
package main

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

func main() {
    f, err := os.Open("a.txt")
    if err != nil {
        fmt.Printf("err: %v\n", err)
    }
    readCloser := ioutil.NopCloser(f)
    fmt.Printf("readCloser: %v\n", readCloser)
}
#结果
readCloser: {0xc000006028}
```

ReadAll

```
func ReadAll(r io.Reader) ([]byte, error)
```

从r读取数据直到EOF或遇到error，返回读取的数据和可能的错误。成功的调用返回的err为nil而非EOF。因为本函数定义为读取r直到EOF，它不会将读取返回的EOF视为应报告的错误。

示例：

```
package main

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

func main() {
    f, _ := os.Open("a.txt") // File实现了Reader 文本内容为hello world
    defer f.Close()

    b, err := ioutil.ReadAll(f)

    if err != nil {
        fmt.Printf("err: %v\n", err)
    }

    fmt.Printf("string(b): %v\n", string(b))
}
#结果
string(b):  hello world
```

ReadDir

```
func ReadDir(dirname string) ([]fs.FileInfo, error)
```

读取dirname目录内的所有文件信息，注意此序列有序。

示例：

```
package main

import (
    "fmt"
    "io/ioutil"
)

func main() {
    fi, _ := ioutil.ReadDir(".")
    for _, v := range fi {
        fmt.Printf("v.Name(): %v\n", v.Name())
    }
}
```

ReadFile

```
func ReadFile(filename string) ([]byte, error)
```

从filename指定的文件中读取数据并返回文件的内容。对err的判断和ReadAll一样。

示例：

```
package main

import (
    "fmt"
    "io/ioutil"
)

func main() {
    b, _ := ioutil.ReadFile("a.txt") // 文本内容为 Hello world
    fmt.Printf("string(b): %v\n", string(b))
}
#结果
string(b):  hello world
```

WriteFile

```
func WriteFile(filename string, data []byte, perm fs.FileMode) error
```

函数向filename指定的文件中写入数据。如果文件不存在将按给出的perm权限创建文件，否则在写入数据之前清空文件。

示例：

```
package main

import (
    "io/ioutil"
)

func main() {
    ioutil.WriteFile("a.txt", []byte("hello world"), 0664)
}
```

TempDir

```
func TempDir(dir, pattern string) (name string, err error)
```

在dir目录里创建一个新的、使用prfix作为前缀的临时文件夹，并返回文件夹的路径。如果dir是空字符串，TempDir使用默认用于临时文件的目录（参见os.TempDir函数）。 不同程序同时调用该函数会创建不同的临时目录，调用本函数的程序有**责任**在**不需要临时文件夹时摧毁**它。

示例：

```
package main

import (
    "fmt"
    "io/ioutil"
    "log"
```

```
    "path/filepath"
)

func main() {
    content := []byte("temporary file's content")
    dir, err := ioutil.TempDir("", "example")
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("dir: %v\n", dir)
    // defer os.RemoveAll(dir) // 销毁临时目录

    tmpfn := filepath.Join(dir, "tmpfile")
    if err := ioutil.WriteFile(tmpfn, content, 0666); err != nil {
        log.Fatal(err)
    }
}
```

TempFile

```
func TempFile(dir, pattern string) (f *os.File, err error)
```

在dir目录下创建一个新的、使用prefix为前缀的临时文件，以读写模式打开该文件并返回os.File指针。如果dir是空字符串，TempFile使用默认用于临时文件的目录（参见os.TempDir函数）。**责任**与TempDir相同。

示例：

```
package main

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

func main() {
    content := []byte("temporary file's content")
    tmpfile, err := ioutil.TempFile("", "example")
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("tmpfile.Name(): %v\n", tmpfile.Name())

    // defer os.Remove(tmpfile.Name()) // 销毁临时文件

    if _, err := tmpfile.Write(content); err != nil {
        log.Fatal(err)
    }
    if err := tmpfile.Close(); err != nil {
        log.Fatal(err)
    }
```

```
}
```