错误处理

目录：

一、错误处理

（一）、错误是什么？

1、错误指程序中出现不正常的情况，从而导致程序无法正常执行。

- 大多语言中使用try...catch...finally语句执行。
- 假设我们正在尝试打开一个文件，文件系统中不存在这个文件。这是一个异常情况，它表示为一个错误。
- 不要忽略错误。永远不要忽略一个错误。忽视错误会招致麻烦。让我重新编写一个示例，该示例列出了与模式匹配的所有文件的名称，而忽略了错误处理代码。

2、Go语言中没有try...catch

- Go 语言通过内置的错误类型提供了非常简单的错误处理机制。
- 错误值可以存储在变量中，通过函数中返回。
- 如果一个函数或方法返回一个错误，按照惯例，它必须是函数返回的最后一个值。
- 处理错误的惯用方式是将返回的错误与nil进行比较。
- nil值表示没有发生错误，而非nil值表示出现错误。
- 如果不是nil，需打印输出错误。

3、error错误类型的本质

- error本质上是一个接口类型，其中包含一个Error()方法。

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

任何实现这个接口的类型都可以作为一个错误使用。这个方法提供了对错误

的描述。

（二）、创建error对象的几种方式

1、errors包下的New()函数返回error对象

- errors.New()创建新的错误。
- 代码分析

```
// Package errors implements functions to manipulate
errors.
package errors

// New returns an error that formats as the given text.
func New(text string) error {
    return &errorString{text}
}

// errorString is a trivial implementation of error.
type errorString struct {
    s string
}

func (e *errorString) Error() string {
    return e.s
}
```

2、fmt包下的Errorf()函数返回error对象

- fmt包下的Errorf()函数本质上还是调用errors.New()

```
// Errorf formats according to a format specifier and returns the string
// as a value that satisfies error.
func Errorf(format string, a ...interface{}) error {
  return errors.New(Sprintf(format, a...))
}
```

3、创建一个自定义错误。

（三）、自定义错误

1、实现步骤

- 1、定义一个结构体，表示自定义错误的类型
- 2、让自定义错误类型实现error接口的方法： Error() string
- 3、定义一个返回error的函数。根据程序实际功能而定。

2、示例代码：

```
package main
import (
  "fmt"
  "time"
)

//1.定义一个结构体，表示自定义错误的类型
type MyError struct {
  When time.Time
  What string
}

//2、自定义错误类型实现error接口的方法： Error() string
func (e *MyError) Error() string {
  return fmt.Sprintf("%v : %v", e.When, e.What)
}

//3.定义一个返回error的函数。求矩形的面积
func getArea(width, length float64) (float64, error) {
  errorMsg := ""
  if width < 0 && length < 0 {
    errorMsg = fmt.Sprintf("长度:%v ， 宽度:%v ， 均为负数", length,
width)
  } else if length < 0 {
    errorMsg = fmt.Sprintf("长度:%v ， 出现负数", length)
  } else if width < 0 {
    errorMsg = fmt.Sprintf("宽度:%v ， 出现负数", width)
```

```
    }

    if errorMsg != "" {
        return 0, &MyError{time.Now(), errorMsg}
    } else {
        return width * length, nil
    }
}

func main() {
    res1, err := getArea(-4, -6)
    if err != nil {
        fmt.Printf(err.Error())
    } else {
        fmt.Println("面积是:", res1)
    }
}
```