**Go语言学习之log包(The way to go)**

讲真，go为我们提供的log package的功能挺简单的，但是常规的功能够用了，我们就进行简简单单的介绍。之后，会自己封装自己log，比如用来记录我们的http日志等。

**import( “log” )**  
Package log implements a simple logging package. It defines a type, Logger, with methods for formatting output. It also has a predefined ‘standard’ Logger accessible through helper functions Print[f|ln], Fatal[f|ln], and Panic[f|ln], which are easier to use than creating a Logger manually. That logger writes to standard error and prints the date and time of each logged message. Every log message is output on a separate line: if the message being printed does not end in a newline, the logger will add one. The Fatal functions call os.Exit(1) after writing the log message. The Panic functions call panic after writing the log message.

**log中的常量标志**

const (

Ldate = 1 << iota *// 2009/01/23*

Ltime *// 01:23:23*

Lmicroseconds *// 01:23:23.123123*

Llongfile *// /a/b/c/d.go:23*

Lshortfile *// d.go:23. overrides Llongfile*

LUTC

LstdFlags = Ldate | Ltime

)

看看几个方法吧：  
**func Fatal**

func Fatal(v ...interface{})

Fatal is equivalent to Print() followed by a call to os.Exit(1).  
例子：

package main

import (

"log"

)

func main(){

log.Fatal("Come with fatal,exit with 1 \n")

}

**func Panic**

func Panic(v ...interface{})

Panic is equivalent to Print() followed by a call to panic().

**func Panicf**

func Panicf(format string, v ...interface{})

Panicf is equivalent to Printf() followed by a call to panic().

**func Panicln**

func Panicln(v ...interface{})

Panicln is equivalent to Println() followed by a call to panic().

**func Print**

func Print(v ...interface{})

Print calls Output to print to the standard logger. Arguments are handled in the manner of fmt.Print.

**func Printf**

func Printf(format string, v ...interface{})

Printf calls Output to print to the standard logger. Arguments are handled in the manner of fmt.Printf.

**func Println**

func Println(v ...interface{})

Println calls Output to print to the standard logger. Arguments are handled in the manner of fmt.Println.

**自定义Logger类型**

type Logger struct {

*// contains filtered or unexported fields*

}

例子：

package main

import (

"bytes"

"fmt"

"log"

)

func main() {

var buf bytes.Buffer

logger := log.New(&buf, "logger: ", log.Lshortfile)

logger.Print("Hello, log file!")

fmt.Print(&buf)

}

**使用例子**  
例子1：

package main

import (

"io"

"io/ioutil"

"log"

"os"

)

var (

Trace \*log.Logger

Info \*log.Logger

Warning \*log.Logger

Error \*log.Logger

)

func Init(

traceHandle io.Writer,

infoHandle io.Writer,

warningHandle io.Writer,

errorHandle io.Writer) {

Trace = log.New(traceHandle,

"TRACE: ",

log.Ldate|log.Ltime|log.Lshortfile)

Info = log.New(infoHandle,

"INFO: ",

log.Ldate|log.Ltime|log.Lshortfile)

Warning = log.New(warningHandle,

"WARNING: ",

log.Ldate|log.Ltime|log.Lshortfile)

Error = log.New(errorHandle,

"ERROR: ",

log.Ldate|log.Ltime|log.Lshortfile)

}

func main() {

Init(ioutil.Discard, os.Stdout, os.Stdout, os.Stderr)

Trace.Println("I have something standard to say")

Info.Println("Special Information")

Warning.Println("There is something you need to know about")

Error.Println("Something has failed")

}

输出：  
INFO: 2017/04/18 00:39:12 log.go:44: Special Information  
WARNING: 2017/04/18 00:39:12 log.go:45: There is something you need to know about  
ERROR: 2017/04/18 00:39:12 log.go:46: Something has failed

例子2：

package main

import (

"log"

"os"

)

func main(){

fileName := "xxx\_debug.log"

logFile,err := os.Create(fileName)

defer logFile.Close()

if err != nil {

log.Fatalln("open file error !")

}

debugLog := log.New(logFile,"[Debug]",log.Llongfile)

debugLog.Println("A debug message here")

debugLog.SetPrefix("[Info]")

debugLog.Println("A Info Message here ")

debugLog.SetFlags(debugLog.Flags() | log.LstdFlags)

debugLog.Println("A different prefix")

}