

日志收集

接口自动化测试课程



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

1. 掌握如何使用logging实现日志收集



目录

Contents

- ◆ 认识日志收集
- ◆ Logging日志模块用法

认识日志收集

01

- 什么是日志?
- 为什么要收集日志?
- 日志级别

学习目标

- 理解在项目中为什么要收集日志
- 知道如何区分日志信息的重要级别

什么是日志?

日志

日志就是用于记录系统运行时的信息，对一个事件的记录；也称为Log

有哪些信息需要记录?

- 脚本运行过程中某个重要变量的值
- 方法的输入参数和返回结果
- 异常信息

为什么要收集日志?



日志级别



系统每天产生的日志信息无法估量，需要一行一行的去分析和记录吗？



系统记录用户操作过程信息时，会设计好日志记录信息的重要性。

日志级别

是指日志信息的优先级、重要性或者严重程度

日志级别	描述
DEBUG	调试级别，打印非常详细的日志信息，通常用于代码调试
INFO	信息级别，一般用于记录突出强调的运行过程步骤
WARNING	警告级别，可能出现潜在错误的情况，一般不影响系统使用
ERROR	错误级别，打印错误异常信息，出现BUG
CRITICAL	严重错误级别，系统可能已经无法运行

提示

- 当为程序指定一个日志级别后，程序会记录所有日志级别大于或等于指定日志级别的日志信息，而不是仅仅记录指定级别的日志信息；
- 一般建议只使用DEBUG、INFO、WARNING、ERROR这四个级别





思考

1. 收集日志有什么作用?
2. 常见的日志级别有哪些?



目录

Contents

◆ 认识日志收集

◆ Logging日志模块用法

02

Logging日志模块用法

学习目标

- 能够把日志输出到控制台和日志文件中

收集日志想要达成的效果

- 可以把日志输出到不同位置
 - 控制台
 - 日志文件（防止日志文件过大，每日生成一个日志文件）
- 记录更加详细的日志信息
 - 打印日志的时间
 - 日志的级别
 - 打印日志的位置
 - 日志内容
- 可以打印不同级别的日志
 - INFO
 - ERROR

日志收集实现

```
import logging.handlers
import time

# ① 创建日志器对象
logger = logging.getLogger()
# ② 设置日志打印级别
logger.setLevel(logging.DEBUG)
# ③ 创建处理器对象
# 输出到控制台
st = logging.StreamHandler()
# 每日生成一个日志文件
fh = logging.handlers.TimedRotatingFileHandler("./a.log",
                                              when='MIDNIGHT',
                                              interval=1,
                                              backupCount=3)

# ④ 创建格式化器
fmt = "%(asctime)s %(levelname)s [%(filename)s(%(funcName)s:%(lineno)d)] - %(message)s"
formatter = logging.Formatter(fmt)
# ⑤ 给处理器设置格式化器
st.setFormatter(formatter)
fh.setFormatter(formatter)
# ⑥ 给日志器添加处理器
logger.addHandler(st)
logger.addHandler(fh)
# ⑦ 打印日志
while True:
    logger.info("test_logger")
    time.sleep(1)
```

日志器介绍



Logger日志器

- 将日志消息传送给所有感兴趣的日志handlers

- 向程序暴露记录日志的方法

创建日志器

```
logger = logging.getLogger()
```

设置Handle

```
logger.addHandler(参数为handler对象)
```

打印日志

```
logger.debug()  
logger.info()  
logger.warning()  
logger.error()  
logger.critical()
```

日志器可以对需要打印的日志设置级别过滤

```
logger.setLevel(logging.DEBUG)
```

Handler处理器介绍



Handler 处理器

Handler对象的作用是将消息分发到handler指定的位置。

logger.addHandler(**参数为handler对象**)

- 输出到哪里添加哪个处理器对象
- 输出到多个地方添加多次处理器

输出-控制台

logging.StreamHandler()

输出-文件

logging.FileHandler()

按时间切割日志
文件

logging.handlers.TimedRotatingFileHandler()

注意

使用切割日志处理器对象时，
需要导包到：logging.handlers

- 输出的日志需要格式，需要添加
格式设置

设置日志格式

handler.setFormatter(**Formatter对象**)

Formatter格式化器介绍



Formatter对象用于配置日志信息的格式

创建格式化器

```
formatter = logging.Formatter(fmt=None)
```

fmt: 指定消息格式化字符串，如果不指定该参数则默认使用message的原始值

handler.setFormatter(**Formatter对象**)



Formatter格式化器介绍



占位符	描述
%(name)s	调试级别，打印非常详细的日志信息，通常用于代码调试
%(levelNo)s	信息级别，一般用于记录突出强调的运行过程步骤
%(levelname)s	警告级别，可能出现潜在错误的情况，一般不影响系统使用
%(pathname)s	错误级别，打印错误异常信息，出现BUG
%(filename)s	严重错误级别，系统可能已经无法运行
%(lineno)d	文本形式的日志级别
%(asctime)s	字符串形式的当前时间。默认格式是“2003-07-08 16:49:45,896”
%(message)s	用户输出的消息
%(funcName)s	调用日志输出函数的函数名

打印日志



调用logging中的不同方法可以打印不同级别的日志

```
import logging

logging.debug("我是一条调试级别日志")
logging.info("我是一条信息级别日志")
logging.warning("我是一条警告级别日志")
logging.error("我是一条错误级别日志")
logging.critical("我是一条严重错误级别日志")
```

注意

打印日志使用的是小写的字母



思考

1. 如何把日志输出到控制台?
2. 如何把日志输出到日志文件?
3. 如何设置日志的格式?
4. 如何打印不同级别的日志?



总结

1. 为什么要收集日志?
2. 如何使用Logging日志模块实现日志的收集?



传智教育旗下高端IT教育品牌