日志配置与管理

hydra 使用lib4go进行本地日志输出与存储,基于rlog进行远程日志存储。

1. 配置本地日志

日志配置文件不需要手动创建，系统启动时会检查 `../conf/logger.json` 文件是否存在，不存在则会自动创建。可手工修改 `../conf/logger.json` 内容，改变日志存储目录和日志输出内容:

`../conf/logger.json` 内容如下:

```
[
  {
    "type": "file",
    "level": "All",
    "path": "/home/colin/work/logs/%date.log",
    "layout": "[%datetime.%ms][%l][%session] %content%n"
  },
  {
    "type": "stdout",
    "level": "All",
    "layout": "[%datetime.%ms][%l][%session]%content"
  }
]
```

参数说明:

参数名	说明
type	日志输出类型，file:文件，stdout：终端
level	日志级别控制开关，值:All < Debug < Info < Warn < Error < Fatal < Off, 日志只显示大于等于配置的级别，即配置为 Info 则，只有 Info,Warn,Error,Fatal 级别的日志才会被记录或输出
path	输出路径，日志输出类型为 file 时必须
layout	输出格式

日志通过 % 开头的变量名进行参数转换:

参数名	说明

参数名	说明
%name	日志名称
%level	日志等级全称
%l	日志等级首字母
%session	当前日志的 sesssion id,每个日志组件创建时都会有一个 session id, 表示相关联的日志信息
%date	日期，格式:20190703
%datetime	日期时间格式，格式:2019/07/03 11:18:07
%yy	年,格式:2019
%mm	月，格式：07
%dd	日期，格式：03
%hh	小时，24 小时制
%mi	分钟，格式：09
%ss	秒数,格式:04
%ms	毫秒，纳秒
%pid	当前进程编号
%caller	日志调用函数名称
%index	日志序号
%ip	当前主机 ip
%n	换行符
%content	日志内容

2. 配置远程日志

hydra 中对 `rlog` 进行了封装，提供了通过 RPC 远程保存日志的功能。

远程日志配置开启后不会影响本地日志记录，可修改 `../conf/logger.json` 关闭本地日志保存。

添加远程日志服务:

```
        app.Conf.Plat.SetVarConf("global", "logger", `{
        "level":"All",
                "interval":"1s",
                "layout":{"name":"%name","time":"%datetime","content":"%content","level"
        "service":"/hydra/log/save@micserver.rlog"
}`)
```

- 添加以上配置后，启动服务器时必须增加 `-l` 或 `-rlog` 参数才会启用远程日志服务

参数说明:

参数名	说明
level	输出的最低日志等级。与 `./conf/logger.json` 配置相同
interval	远程保存时间间隔
layout	输出格式
service	`rlog` 远程服务名称

> 当前配置会将 1 秒内的所有日志压缩后提交给 `rlog` 服务器， `rlog` 服务器收到后解压并存储到 `elastic search` 。

3. 创建日志对象

- `github.com/micro-plat/lib4go/logger` 包初始化时即会加载 `../conf/logger.json` 文件，并生成全局日志输出对象。
- 通过 `logger.New` , `logger.GetSession` 可创建应用自己的日志组件 `*logger.Logger` 。此组件只会保存 `日志名称` ， `session id` , `tags` 。并且使用 `sync.Pool` 对日志进行缓存管理。频繁创建的日志组件，不再使用时应使用 `*logger.Logger` 的 `Close` 方法进行组件回收，减少内存占用。
- 由于日志是异步输出，保证所有日志都能全部输出，请在系统退出前调用 `logger.Close()`
- 通过 `*logger.Logger` 组件可打印 Debug, Info , Warn , Error , Fatal 级别的日志。日志不会同步写入文件与终端，而是放入 channel 缓冲区，根据一定的策略写入文件或终端。对于 linux,mac 不同日志级别会显示不同的颜色。
- `logger.New` , `logger.GetSession` 都可创建 `*logger.Logger` 对象，区别是 `logger.New` 生成新的 `session id` , `logger.GetSession` 需传入用户定义的 `session id`

日志组件使用示例:

```
log:=logger.New("apiserver")
log.Debug("启动apiserver") //输出蓝色
log.Info("启动apiserver") //输出绿色
log.Warn("启动apiserver") //输出黄色
log.Error("启动apiserver") //输出红色
```