Grails3 配置 logback 日志

描述：

本文讲述如何配置根据每天生成单独的日志文件以及日志格式和日志文件大小进行配置。

打开项目目录下的 grails-app/conf/logback.groovy

```
import ch.qos.logback.classic.Level

import ch.qos.logback.classic.filter.LevelFilter

import ch.qos.logback.core.spi.FilterReply


// See http://logback.qos.ch/manual/groovy.html for details on configuration

appender('STDOUT', ConsoleAppender) {

    encoder(PatternLayoutEncoder) {

        pattern = "%-4(%d{HH:mm:ss.SSS} [%thread]) %-5level %logger{32} \\(%file:%line\\) - %msg%n"

    }

}

logger("console",ERROR,['STDOUT'],false)

root(ERROR, ['STDOUT'])

//日志级别从高到地低 OFF 、 FATAL 、 ERROR 、 WARN 、 INFO 、 DEBUG 、 TRACE 、 ALL


//定义当前目录

def HOME_DIR = "."


//ERROR 级别的日志

appender("ERROR", RollingFileAppender) {

    //过滤器，只记录 ERROR 级别的日志

    filter(LevelFilter) {

        level = Level.ERROR

        onMatch = FilterReply.ACCEPT

        onMismatch = FilterReply.DENY

    }

    //PatternLayoutEncoder 对输出日志信息进行格式化

    encoder(PatternLayoutEncoder) {

        //默认为 pattern = "%level %logger - %msg%n"

        //%d 表示日期，%thread 表示线程名，%level 日志级别，%file 具体的文件，%line 记录日志位置，%msg 日志消息，%n 换行符

//        pattern = "%-4(%d{HH:mm:ss.SSS} [%thread]) %-5level %logger{32} \\(%file:%line\\) - %msg%n"

        pattern = "%-4(%d{HH:mm:ss.SSS} [%thread]) %-5level %logger{32} \\(%F:%L\\) - %msg%n"

    }

    //指定日志生成格式

    rollingPolicy(TimeBasedRollingPolicy) {

        fileNamePattern = "${HOME_DIR}/logs/%d{yyyy-MM-dd}_ERROR_%i.log"

        maxHistory = 30 //日志最长保留 30 天

        timeBasedFileNamingAndTriggeringPolicy(SizeAndTimeBasedFNATP) {

            maxFileSize = "10MB"
```

```
        }
    }


    //约束生成单个日志文件大小为 10M，超过后新建一个文件保存
//    triggeringPolicy(SizeBasedTriggeringPolicy) {
//        maxFileSize = "10MB"
//    }
    append = true
}
logger("console",ERROR,['ERROR'],false)


//INFO 级别的日志
appender("INFO", RollingFileAppender) {
    //过滤器，只记录 INFO 级别的日志
    filter(LevelFilter) {
        level = ch.qos.logback.classic.Level.INFO
        onMatch = FilterReply.ACCEPT
        onMismatch = FilterReply.DENY
    }
    //PatternLayoutEncoder 对输出日志信息进行格式化
    encoder(PatternLayoutEncoder) {
        //默认为 pattern = "%level %logger - %msg%n"
        //%d 表示日期，%thread 表示线程名，%level 日志级别，%file 具体的文件，%line 记录日志位置，%msg 日志消息，%n 换行符
        pattern = "%-4(%d{HH:mm:ss.SSS} [%thread]) %-5level %logger{32} \\(%file:%line\\) - %msg%n"
    }
    //指定日志生成格式，文件名以日期命名，生成每日日志文件，如果超出大小则另起文件存放
    //%d{yyyy-MM-dd}-日期，%i-用于记录每日日志个数
    rollingPolicy(TimeBasedRollingPolicy) {
        fileNamePattern = "${HOME_DIR}/logs/%d{yyyy-MM-dd}_INFO_%i.log"
        maxHistory = 30 //日志最长保留 30 天
        timeBasedFileNamingAndTriggeringPolicy(SizeAndTimeBasedFNATP) {
            maxFileSize = "10MB" //单个日志文件最大为 10MB
        }
    }
//    //约束生成单个日志文件大小为 10M，超过后新建一个文件保存
//    triggeringPolicy(SizeBasedTriggeringPolicy) {
//        maxFileSize = "10MB"
//    }
    append = true
}
//为 false 表示只打印到本 log 的 appender 中，不打印父类
logger("console",INFO,['INFO'],false)


//将指定级别的日志输出到日志文件中
```

```
root(ERROR, ['ERROR'])

//root(INFO, ['INFO'])
```

在需要调用的地方直接调用,在类上面添加注解@groovy.util.logging.Slf4j 用于记录调用日志的位置

```
@Slf4j

class IndexController {

   def index() {

       def info = [result:false,msg:"操作失败"]

       log .error("参数信息: error={}" , info)

   }

}
```

效果图：

```
 17:15:42.040 [http-nio-8080-exec-10] ERROR com.system.IndexController (IndexController.groovy:10) - 参数信息: error={result=false, msg=操作失败}
```