# 一、基础概念

## 1. SpringBoot中的日志

Logback是log4j框架的作者开发的新一代日志框架，它效率更高、能够适应诸多的运行环境，同时天然支持SLF4J。默认情况下，Spring Boot会用Logback来记录日志，并用INFO级别输出到控制台。

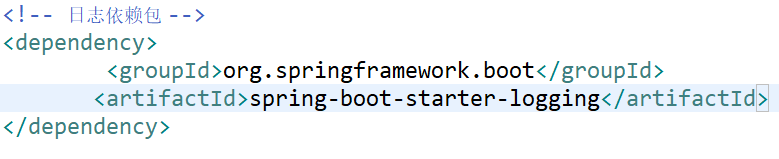
## 2. 日志输出的内容格式

1. 时间日期：精确到毫秒
2. 日志级别：ERROR, WARN, INFO, DEBUG or TRACE
3. 进程ID
4. 分隔符：--- 标识实际日志的开始
5. 线程名：方括号括起来（可能会截断控制台输出）
6. Logger名：通常使用源代码的类名
7. 日志内容

# 二、具体配置

## 1. 默认属性配置

Spring Boot为我们提供了很多默认的日志配置，所以，只要将spring-boot-starter-logging作为依赖加入到当前应用的classpath，则“开箱即用”。



## 2. 控制台输出

1. 日志级别从低到高分为TRACE < DEBUG < INFO < WARN < ERROR < FATAL，如果设置为WARN，则低于WARN的信息都不会输出。Spring Boot中默认配置ERROR、WARN和INFO级别的日志输出到控制台。
2. **在application.properties中配置debug=true**，该属性置为true的时候，核心Logger（包含嵌入式容器、hibernate、spring）会输出更多内容，但是你自己应用的日志并不会输出为DEBUG级别。

## 3. 文件输出

1. 默认情况下，Spring Boot将日志输出到控制台，不会写到日志文件。如果要编写除控制台输出之外的日志文件，则需在application.properties中设置logging.file或logging.path属性。
2. logging.file，设置文件，可以是绝对路径，也可以是相对路径。如：logging.file=my.log
3. logging.path，设置目录，会在该目录下创建spring.log文件，并写入日志内容，如：logging.path=/var/log
4. 如果只配置 logging.file，会在项目的当前路径下生成一个 xxx.log 日志文件。

如果只配置 logging.path，在 /var/log文件夹生成一个日志文件为 spring.log

二者不能同时使用，如若同时使用，则只有logging.file生效

1. 默认情况下，日志文件的大小达到10MB时会切分一次，产生新的日志文件，默认级别为：ERROR、WARN、INFO

## 4. 级别控制

1. 所有支持的日志记录系统都可以在Spring环境中设置记录级别（例如在application.properties中）

格式为：’logging.level.\* = LEVEL’

logging.level：日志级别控制前缀，\*为包名或Logger名

LEVEL：选项TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF

1. 实例

logging.level.com.dudu=DEBUG：com.dudu包下所有class以DEBUG级别输出

logging.level.root=WARN：root包下的日志以WARN级别输出

## 5. 自定义日志配置

1. 由于日志服务一般都在ApplicationContext创建前就初始化了，它并不是必须通过Spring的配置文件控制。因此通过系统属性和传统的Spring Boot外部配置文件依然可以很好的支持日志控制和管理。
2. **根据不同的日志系统，你可以按如下规则组织配置文件名，就能被正确加载**
3. Logback：logback-spring.xml, logback-spring.groovy, logback.xml, logback.groovy
4. Log4j：log4j-spring.properties, log4j-spring.xml, log4j.properties, log4j.xml
5. Log4j2：log4j2-spring.xml, log4j2.xml
6. JDK (Java Util Logging)：logging.properties
7. Spring Boot官方推荐优先使用带有-spring的文件名作为你的日志配置（如使用logback-spring.xml，而不是logback.xml），命名为logback-spring.xml的日志配置文件，spring boot可以为它添加一些spring boot特有的配置项（下面会提到）。

上面是默认的命名规则，并且放在src/main/resources下面即可。

1. 如果你即想完全掌控日志配置，但又不想用logback.xml作为Logback配置的名字，可以在application.properties配置文件里面通过logging.config属性指定自定义的名字

C:\Users\HP\AppData\Local\Temp\1514875204(1).png

1. 具体配置详情 -- 见文件logback-spring.xml

## 6. 多环境日志输出

1. 据不同环境（prod:生产环境，test:测试环境，dev:开发环境）来定义不同的日志输出，在 logback-spring.xml中使用 springProfile 节点来定义，方法如下：

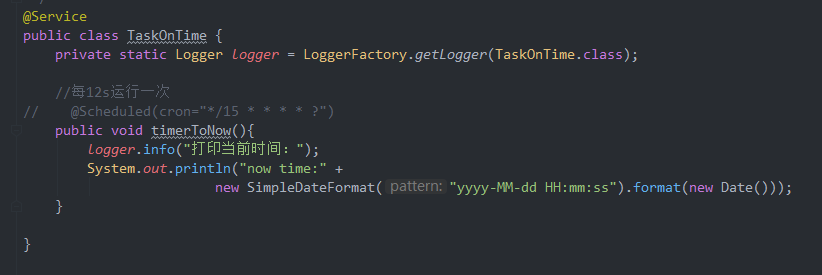
文件名称不是logback.xml，想使用spring扩展profile支持，要以logback-spring.xml命名



## 7. Logback的使用

7.1 在代码中插入日志

如：private static Logger logger = LoggerFactory.getLogger(TashOnTime.class);



# 三、 日志框架的区别

## 1. SLF4J - Simple Logging Facade for Java.

(1) SLF4J，即简单日志门面（Simple Logging Facade for Java），不是具体的日志解决方案，而是通过Facade Pattern提供一些Java logging API，它只服务于各种各样的日志系统。

(2) 实际上，SLF4J所提供的核心API是一些接口以及一个LoggerFactory的工厂类。在使用SLF4J的时候，不需要在代码中或配置文件中指定你打算使用那个具体的日志系统。SLF4J提供了统一的记录日志的接口，只要按照其提供的方法记录即可，最终日志的格式、记录级别、输出方式等通过具体日志系统的配置来实现，因此可以在应用中灵活切换日志系统。

## 2. JUI - java.util.logging.

(1) Java提供了自己的日志框架，类似于Log4J，但是API并不完善，对开发者不是很友好，而且对于日志的级别分类也不是很清晰，比如：SEVERE, WARNING, INFO, CONFIG, FINE,FINER, FINEST。所以不推荐使用这种方式输出日志。

## 3. JCL - Jakarta Commons-Logging.

(1) Jakarta Commons Logging和SLF4J非常类似，也是提供的一套API来掩盖了真正的Logger实现。便于不同的Logger的实现的替换，而不需要重新编译代码。缺点在于它的查找Logger的实现者的算法比较复杂，而且当出现了一些class loader之类的异常时，无法去修复它。

## 4. Log4j - Log for Java.

(1) Log4j是Apache的一个开放源代码项目，通过使用Log4j，我们可以控制日志信息输送的目的地是控制台、文件、数据库等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

1. Log4j有7种不同的log级别，按照等级从低到高依次为：

TRACE<DEBUG<INFO<WARN<ERROR<FATAL<OFF。如果配置为OFF级别，表示关闭log。

1. Log4j支持两种格式的配置文件：properties和xml。包含三个主要的组件：Logger、appender、Layout。

## 5. Log4j2

(1) 对Log4j框架的改良版本。、

(2) 插件式结构。Log4j 2支持插件式结构。我们可以根据自己的需要自行扩展Log4j 2. 我们可以实现自己的appender、logger、filter。

(3) 配置文件优化。在配置文件中可以引用属性，还可以直接替代或传递到组件。而且支持json格式的配置文件。不像其他的日志框架，它在重新配置的时候不会丢失之前的日志文件。

## 6. Logback

6.1 简介

(1) Logback，一个“可靠、通用、快速而又灵活的Java日志框架”。logback当前分成三个模块：logback-core，logback- classic和logback-access。logback-core是其它两个模块的基础模块。logback-classic是log4j的一个改良版本。此外logback-classic完整实现SLF4J API使你可以很方便地更换成其它日志系统如log4j或JDK14 Logging。logback-access访问模块与Servlet容器集成提供通过Http来访问日志的功能。

6.2 优点

1. logback比log4j要快大约10倍，而且消耗更少的内存。
2. logback-classic模块直接实现了SLF4J的接口，所以我们迁移到logback几乎是零开销的。
3. logback不仅支持xml格式的配置文件，还支持groovy格式的配置文件。相比之下，Groovy风格的配置文件更加直观，简洁。
4. logback-classic能够检测到配置文件的更新，并且自动重新加载配置文件。
5. logback能够优雅的从I/O异常中恢复，从而我们不用重新启动应用程序来恢复logger。
6. logback能够根据配置文件中设置的上限值，自动删除旧的日志文件。
7. logback能够自动压缩日志文件。
8. logback能够在配置文件中加入条件判断（if-then-else)。可以避免不同的开发环境（dev、test、uat...）的配置文件的重复。
9. logback带来更多的filter。
10. logback的stack trace中会包含详细的包信息。
11. logback-access和Jetty、Tomcat集成提供了功能强大的HTTP-access日志。