使用Logback日志框架。

Spring Boot在所有内部日志中使用Commons Logging，但是默认配置也提供了对常用日志的支持，如：Java Util Logging，Log4j，Log4j2和Logback。每种Logger都可以通过配置使用控制台或者文件输出日志内容。

### 1、默认日志Logback

SLF4J——Simple Logging Façade For Java，它是一个针对于各类Java日志框架的统一Facade抽象。SLF4J定义了统一的日志抽象接口，而真正的日志实现则是在运行时决定的——它提供了各类日志框架的binding。

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。在运行应用程序和其他例子时，你应该已经看到很多INFO级别的日志了。



日志输出内容元素具体如下：

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

#### 1.1添加日志依赖

假如maven依赖中添加了spring-boot-starter-logging：

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

那么将自动使用Logback作为应用日志框架，Spring Boot启动的时候，由org.springframework.boot.logging.Logging-Application-Listener根据情况初始化并使用。

实际开发中我们不需要直接添加该依赖，你会发现spring-boot-starter其中包含了spring-boot-starter-logging，该依赖内容就是Spring Boot默认的日志框架L

Logback。

#### 1.2默认配置属性支持

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”。

下面介绍几种在application.properties就可以配置的日志相关属性。

#### 1.3控制台输出

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。您还可以通过启动您的应用程序-debug标志来启用“调试”模式（开发的时候推荐开启），以下两种方式皆可：

1. 在运行命令后加入—debug标志，如：$ java -jar springTest.jar –debug；
2. 在application.properties中配置debug=true，该属性设置为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。

#### 1.4文件输出

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（两者不可同时使用）。

logging.file，设置文件，可以是绝对路径，也可以是相对路径。

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

如果只配置logging.file，会在项目的当前路径下生成一个xxx.log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

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

#### 1.5级别控制

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：

logging.level.\*=LEVEL

logging.level日志级别控制前缀，\*为包名或Logger名。

例如：

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

logging.level.root=WARN root日志以WARN级别输出。

#### 1.6自定义日志配置

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的外部配置文件依然可以很好的支持日志控制和管理。

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：

* Logback：logback-spring.xml，logback-spring.groovy，logback.xml，logback.grovvy
* Log4j：log4j-spring.properties，log4j-spring.xml，log4j.properties，log4j.xml
* Log4j2：log4j2-spring.xml，log4j2.xml
* JUL（Java Util Logging）：logging.properties

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。配置文件放在src/main/resources下面即可。

### 2、logback.xml

#### 2.1基本配置

如果你既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：

logging.config=classpath:logging-config.xml

虽然一般并不需要改变配置文件的名字，但是如果你想针对不同运行时Profile使用不同的日志配置。

logback-spring.xml例子：

<?xml version="1.0" encoding="UTF-8"?>

<configuration scan="true" scanPeriod="60 seconds" debug="false">

<contextName>logback</contextName>

<property name="log.path" value="/Users/tengjun/Documents/log" />

<!--输出到控制台-->

<appender name="console" class="ch.qos.logback.core.ConsoleAppender">

<!-- <filter class="ch.qos.logback.classic.filter.ThresholdFilter">

<level>ERROR</level>

</filter>-->

<encoder>

<pattern>%d{HH:mm:ss.SSS} %contextName [%thread] %-5level %logger{36}.%M:%L - %msg%n</pattern>

</encoder>

</appender>

<!--输出到文件-->

<appender name="file" class="ch.qos.logback.core.rolling.RollingFileAppender">

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>${log.path}/logback.%d{yyyy-MM-dd}.log</fileNamePattern>

</rollingPolicy>

<encoder>

<pattern>%d{HH:mm:ss.SSS} %contextName [%thread] %-5level %logger{36}.%M:%L - %msg%n</pattern>

</encoder>

</appender>

<root level="info">

<appender-ref ref="console" />

<appender-ref ref="file" />

</root>

<!-- logback为java中的包 -->

<logger name="com.dudu.controller"/>

<!--logback.LogbackDemo：类的全路径 -->

<logger name="com.dudu.controller.LearnController" level="WARN" additivity="false">

<appender-ref ref="console"/>

</logger>

</configuration>

#### 2.1属性分析

（1）根节点

（2）子节点一<appender>

（3）输出到文件RollingFileAppender

（4）子节点二

（5）子节点三

#### 2.3多环境日志输出

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：

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<include resource="org/springframework/boot/logging/logback/base.xml" />

<logger name="org.springframework.web" level="INFO"/>

<logger name="org.springboot.sample" level="TRACE" />

<!-- 开发、测试环境，多个用逗号隔开 -->

<springProfile name="dev,test">

<logger name="org.springframework.web" level="INFO"/>

<logger name="org.springboot.sample" level="INFO" />

<logger name="com.wuwii" level="DEBUG" additivity="false" />

</springProfile>

<!-- 生产环境 -->

<springProfile name="pro">

<logger name="org.springframework.web" level="ERROR"/>

<logger name="org.springboot.sample" level="ERROR" />

<logger name="com.wuwii" level="ERROR" additivity="false" />

</springProfile>

</configuration>

可以启动服务的时候指定 profile （如不指定使用默认），如指定prod 的方式为：

java -jar xxx.jar –spring.profiles.active=prod

#### 2.4格式化颜色

注意到Spring Boot输出日志到控制台按日志级别有不同的颜色，logback官方给出了解释：

<https://logback.qos.ch/manual/layouts.html#coloring>

<encoder>

<pattern>%d{HH:mm:ss.SSS} %contextName

[%thread] %highlight(%-5level) %cyan(%logger{36}).%M:%L - %msg%n</pattern>

</encoder>