4、日志

2020年5月19日 10:08

- 1、日志框架
  - 市面上的日志框架：

    JUL、JCL、Jboss-logging、logback、log4j、log4j2、slf4j...

日志门面 （日志的抽象层）	日志实现
JCL（Jakarta Commons Logging） SLF4j（Simple Logging Facade for Java） jboss-logging	Log4j JUL（java.util.logging） Log4j2 Logback

    SpringBoot选用 SLF4j和logback；
- 2、SLF4j使用
  - 1、如何在系统中使用SLF4j

    日志记录方法调用，不应该直接调用实现类，而是调用抽象层的方法；

    导入slf4j的jar和logback的实现jar

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class HelloWorld {
  public static void main(String[] args) {
    Logger logger = LoggerFactory.getLogger(HelloWorld.class);
    logger.info("Hello World");
  }
}
```

    每一个日志的实现框架都有自己的配置文件。使用slf4j以后，配置文件还是做成日志实现框架自己本身的配置文件；
  - 2、遗留问题

    统一日志记录，即使是别的框架和我一起统一使用slf4j进行输出？

    如何让系统中所有的日志都统一到slf4j：

    a. 将系统中其他日志框架先排除出去；

    b. 用中间包来替换原有的日志框架；

    c. 我们导入slf4j其他的实现
  - 3、日志使用
    - 1、默认配置

      SpringBoot默认帮我们配置好了日志

```
//记录器
Loggerlogger=LoggerFactory.getLogger(getClass());
@Test
voidcontextLoads(){
/*
*日志级别
*由低到高trace<debug<info<warn<error
*可以调整输出日志级别；日志就只会在这个级别以后的高级级别生效
```

```
**/
logger.trace("这是trace日志。。。");
logger.debug("这是debug日志。。。");
//SpringBoot默认给我们使用的是info级别的，没有指定级别的就用SpringBoot默认规定的级别；root级别
logger.info("这是info日志。。。");
logger.warn("这是warn日志。。。");
logger.error("这是warn日志。。。");
}
```

SpringBoot修改日志的默认配置

```
logging.level.com.lhq=trace
#不指定路径在当前项目下生成springboot.log日志
#可以指定完整路径
#logging.file.name=springboot.log
#在当前磁盘的bilili-springboot文件夹里创建log文件夹；使用spring.log作为默认文件
logging.file.path=/bilili-springboot/log
#在控制台输出的日志的格式
logging.pattern.console=%d{yyyy-MM-dd}[%thread]%-5level%logger{50}-%msg%n
#指定文件中日志输出格式
logging.pattern.file=%d{yyyy-MM-dd}===[%thread]===%-5level===%logger{50}====%msg%n
```

logging.file	logging.path	Example	Description
(none)	(none)		只在控制台输出
指定文件名	(none)	my.log	输出日志到my.log文件
(none)	指定目录	/var/log	输出到指定目录的 spring.log 文件中

日志输出格式：

%d表示日期时间，
%thread表示线程名，
%-5level：级别从左显示5个字符宽度
%logger{50} 表示logger名字最长50个字符，否则按照句点分割。
%msg：日志消息，
%n是换行符

```
%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{50} - %msg%n
```

- 2、指定配置

给类路径下放上每个日志框架自己的配置文件即可；SpringBoot就不使用他默认配置的了

Logging System	Customization
Logback	logback-spring.xml, logback-spring.groovy, logback.xml or logback.groovy
Log4j2	log4j2-spring.xml or log4j2.xml

JDK (Java Util Logging)	logging.properties

logback.xml：直接就被日志框架识别了；

logback-spring.xml：日志框架就不直接加载日志的配置项，由SpringBoot解析日志配置，可以使用SpringBoot的高级Profile功能

```
<springProfilename="dev">//可以指定某段配置只在某个环境下生效
<pattern>%d{yyyy-MM-ddHH:mm:ss.SSS}---》[%thread]---》%-5level%logger{50}-%msg%
n</pattern>
</springProfile>
<springProfilename="!dev">
<pattern>%d{yyyy-MM-ddHH:mm:ss.SSS}---[%thread]---%-5level%logger{50}-%msg%
n</pattern>
</springProfile>
```

如果使用logback.xml作为日志配置文件，还要使用profile功能，会有以下错误
no applicable action for [springProfile]

- 5、切换日志框架

  可以按照slf4j的日志适配图，进行相关的切换；

  slf4j+log4j的方式；

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <artifactId>logback-classic</artifactId>
      <groupId>ch.qos.logback</groupId>
    </exclusion>
    <exclusion>
      <artifactId>log4j-over-slf4j</artifactId>
      <groupId>org.slf4j</groupId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
</dependency>
```

  切换为log4j2

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
          <exclusion>
            <artifactId>spring-boot-starter-logging</artifactId>
            <groupId>org.springframework.boot</groupId>
          </exclusion>
```

```
            </exclusions>
        </dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-log4j2</artifactId>
</dependency>
```