在线oj项目-日志框架

日志框架引入（slf4j+logback）

简介

- **重要性：**
  - 故障的排查和问题定位
  - 系统监控
  - 数据采集
  - 日志审计
- **注意事项：**
  - 注意日志级别
  - 注意日志内容，日志格式和可读性
  - 避免过度日志记录
  - 注意日志的滚动和归档
- **为什么选择slf4j+logback**
  - 易于切换
  - 配置灵活
  - logback性能更好，集成更方便，功能更强大
  - SpringBoot 默认的日志框架

配置文件

- **logback.xml配置：以system为例：**

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true" scanPeriod="60 seconds" debug="false">
    <!-- 日志存放路径 -->
    <property name="log.path" value="logs/oj-system" />
   <!-- 日志输出格式 -->
    <property name="log.pattern" value="%d{HH:mm:ss.SSS} [%thread] %-5level
%logger{20} - [%method,%line] - %msg%n" />

    <!-- 控制台输出 -->
```

```
    <appender name="console" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>${log.pattern}</pattern>
        </encoder>
    </appender>

    <!-- 系统日志输出 -->
    <appender name="file_info" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${log.path}/info.log</file>
        <!-- 循环政策：基于时间创建日志文件 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!-- 日志文件名格式 -->
            <fileNamePattern>${log.path}/info.%d{yyyy-MM-dd}.log</fileNamePattern>
            <!-- 日志最大的历史 10天 -->
            <maxHistory>10</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>${log.pattern}</pattern>
        </encoder>
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <!-- 过滤的级别 -->
            <level>INFO</level>
            <!-- 匹配时的操作：接收（记录） -->
            <onMatch>ACCEPT</onMatch>
            <!-- 不匹配时的操作：拒绝（不记录） -->
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>

    <appender name="file_error" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${log.path}/error.log</file>
        <!-- 循环政策：基于时间创建日志文件 -->
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!-- 日志文件名格式 -->
            <fileNamePattern>${log.path}/error.%d{yyyy-MM-dd}.log</fileNamePattern>
            <!-- 日志最大的历史 10天 -->
            <maxHistory>10</maxHistory>
        </rollingPolicy>
        <encoder>
            <pattern>${log.pattern}</pattern>
        </encoder>
```

```
        <filter class="ch.qos.logback.classic.filter.LevelFilter">
            <!-- 过滤的级别 -->
            <level>ERROR</level>
          <!-- 匹配时的操作：接收（记录） -->
            <onMatch>ACCEPT</onMatch>
          <!-- 不匹配时的操作：拒绝（不记录） -->
            <onMismatch>DENY</onMismatch>
        </filter>
    </appender>

    <!--日志级别-->
    <root level="info">
        <appender-ref ref="file_info" />
        <appender-ref ref="console" />
         <appender-ref ref="file_error" />
    </root>

</configuration>
```

- **核心配置含义：**
  - 鉴于大家已有一定的基础，上述配置的核心部分已做出解释，无需过多赘述。建议大家在后续自行进行充分的测试与练习，以加深理解。
  - 日志输出格式，下面给出了一些常用的日志格式转换词。更多说明, 参考: https://logback.qos.ch/manual/layouts.html#conversionWord

```
%d：日期和时间，可以使用各种格式。在上面的例子中，它使用了 HH:mm:ss.SSS 格式，表示小时、分钟、秒和毫秒。
%thread：产生日志事件的线程名。
%level：日志级别（如 INFO，DEBUG，ERROR 等）。
%logger：产生日志事件的 logger 名，通常用于标识发出日志请求的类或模块。在上面的例子中，%logger{20} 表示 logger 名的最大长度为 20 个字符。
%msg：日志消息，即实际记录的日志内容。
%method表示产生日志事件的方法名
%line表示产生日志事件的行号
```

- **滚动策略：**
  - TimeBasedRollingPolicy：最常用的滚动策略，它根据时间来制定滚动策略。
  - SizeBasedTriggeringPolicy：基于日志文件大小的滚动策略。当日志文件达到指定的大小时，它会被滚动（即创建一个新的日志文件）。
  - FixedWindowRollingPolicy：固定窗口滚动策略。它根据一个固定的窗口大小（即可以保留的日志文件数量）来滚动日志文件。

测试

```java
package com.bite.system.controller;

import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/test")
@Slf4j
public class TestController {

    @GetMapping
    public String test() {
        System.out.println("我是system服务");
        log.info("我是system服务info日志");
        log.error("我是system服务error日志");
        return "我是system服务";
    }
}
```