

Spring Boot集成ELK

参考链接:

https://github.com/qos-ch/logback/tree/v_1.2.3/logback-examples

<https://github.com/logfellow/logstash-logback-encoder/tree/logstash-logback-encoder-6.6>

1 分场景配置

为了方便我们查看日志，我们把日志分为以下四种：

- 调试日志：最全日志，包含了应用中所有 `DEBUG` 级别以上的日志，仅在开发、测试环境中开启收集；
- 错误日志：只包含应用中所有 `ERROR` 级别的日志，所有环境只都开启收集；
- 业务日志：在我们应用 对应包下 打印的日志，可用于查看我们自己在应用中打印的业务日志；
- 记录日志：每个接口的 访问记录，可以用来查看接口执行效率，获取接口访问参数。

下面来看一个完整的配置示例，配置文件命名为 `logback-spring.xml`，放在 `classpath` 下（`resources`根目录）。

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE configuration>
3 <configuration>
4     <!-- 引用默认日志配置 -->
5     <include
6         resource="org/springframework/boot/logging/logback/defaults.xml"/>
7     <!-- 使用默认的控制台日志输出实现 -->
8     <include resource="org/springframework/boot/logging/logback/console-
9         appender.xml"/>
10    <!-- 日志文件保存路径 -->
11    <property name="LOG_FILE_PATH"
12        value="${LOG_FILE:-${LOG_PATH:-${LOG_TEMP:-${java.io.tmpdir:-/tmp}}}/logs}
13    "/>
14    <!-- 项目名称 -->
15    <springProperty name="PROJ_NAME" scope="context"
16        source="application.title" defaultValue="01-star"/>
17    <!-- 应用名称 -->
18    <springProperty name="APP_NAME" scope="context"
19        source="spring.application.name" defaultValue="spring-
20        boot"/>
21    <!-- LogStash访问host -->
22    <springProperty name="LOG_STASH_HOST" scope="context"
23        source="logstash.host" defaultValue="localhost"/>
24
25    <!-- DEBUG日志输出到文件 -->
26    <appender name="FILE_DEBUG"
27        class="ch.qos.logback.core.rolling.RollingFileAppender">
28        <!-- 输出DEBUG以上级别日志 -->
29        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
30            <level>DEBUG</level>
31        </filter>
```

```

28         <encoder>
29             <!-- 设置为默认的文件日志格式 -->
30             <pattern>${FILE_LOG_PATTERN}</pattern>
31             <charset>UTF-8</charset>
32         </encoder>
33         <rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
34             <!-- 设置文件命名格式 -->
35             <fileNamePattern>${LOG_FILE_PATH}/debug/${APP_NAME}-%d{yyyy-MM-
dd}-%i.log</fileNamePattern>
36             <!-- 设置日志文件大小，超过就重新生成文件，默认10M -->
37             <maxFileSize>${LOG_FILE_MAX_SIZE:-10MB}</maxFileSize>
38             <!-- 日志文件保留天数，默认30天 -->
39             <maxHistory>${LOG_FILE_MAX_HISTORY:-30}</maxHistory>
40         </rollingPolicy>
41     </appender>
42
43     <!-- ERROR日志输出到文件 -->
44     <appender name="FILE_ERROR"
45         class="ch.qos.logback.core.rolling.RollingFileAppender">
46         <!-- 只输出ERROR级别的日志 -->
47         <filter class="ch.qos.logback.classic.filter.LevelFilter">
48             <level>ERROR</level>
49             <onMatch>ACCEPT</onMatch>
50             <onMismatch>DENY</onMismatch>
51         </filter>
52         <encoder>
53             <!-- 设置为默认的文件日志格式 -->
54             <pattern>${FILE_LOG_PATTERN}</pattern>
55             <charset>UTF-8</charset>
56         </encoder>
57         <rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
58             <!-- 设置文件命名格式 -->
59             <fileNamePattern>${LOG_FILE_PATH}/error/${APP_NAME}-%d{yyyy-MM-
dd}-%i.log</fileNamePattern>
60             <!-- 设置日志文件大小，超过就重新生成文件，默认10M -->
61             <maxFileSize>${LOG_FILE_MAX_SIZE:-10MB}</maxFileSize>
62             <!-- 日志文件保留天数，默认30天 -->
63             <maxHistory>${LOG_FILE_MAX_HISTORY:-30}</maxHistory>
64         </rollingPolicy>
65     </appender>
66
67     <!-- DEBUG日志输出到LogStash -->
68     <appender name="LOG_STASH_DEBUG"
class="net.logstash.logback.appender.LogstashTcpSocketAppender">
69         <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
70             <level>DEBUG</level>
71         </filter>
72         <destination>${LOG_STASH_HOST}:4560</destination>
73         <encoder charset="UTF-8"
class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">
74             <providers>
75                 <timestamp>
76                 <timeZone>Asia/Shanghai</timeZone>

```

```

77         </timestamp>
78         <!-- 自定义日志输出格式 -->
79         <pattern>
80             <pattern>
81                 {
82                     "project": "${PROJ_NAME:-}",
83                     "level": "%level",
84                     "service": "${APP_NAME:-}",
85                     "pid": "${PID:-}",
86                     "thread": "%thread",
87                     "class": "%logger",
88                     "message": "%message",
89                     "stack_trace": "%exception{20}"
90                 }
91             </pattern>
92         </pattern>
93     </providers>
94 </encoder>
95 <!-- 当有多个Logstash服务时，设置访问策略为轮询 -->
96 <connectionStrategy>
97     <roundRobin>
98         <connectionTTL>5 minutes</connectionTTL>
99     </roundRobin>
100 </connectionStrategy>
101 </appender>
102
103 <!-- ERROR日志输出到Logstash -->
104 <appender name="LOG_STASH_ERROR"
105     class="net.logstash.logback.appender.LogstashTcpSocketAppender">
106     <filter class="ch.qos.logback.classic.filter.LevelFilter">
107         <level>ERROR</level>
108         <onMatch>ACCEPT</onMatch>
109         <onMismatch>DENY</onMismatch>
110     </filter>
111     <destination>${LOG_STASH_HOST}:4561</destination>
112     <encoder charset="UTF-8"
113     class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">
114         <providers>
115             <timestamp>
116                 <timeZone>Asia/Shanghai</timeZone>
117             </timestamp>
118             <!-- 自定义日志输出格式 -->
119             <pattern>
120                 <pattern>
121                     {
122                         "project": "${PROJ_NAME:-}",
123                         "level": "%level",
124                         "service": "${APP_NAME:-}",
125                         "pid": "${PID:-}",
126                         "thread": "%thread",
127                         "class": "%logger",
128                         "message": "%message",
129                         "stack_trace": "%exception{20}"
130                     }
131                 </pattern>

```

```

130         </pattern>
131     </providers>
132 </encoder>
133 <!-- 当有多个LogStash服务时，设置访问策略为轮询 -->
134 <connectionStrategy>
135     <roundRobin>
136         <connectionTTL>5 minutes</connectionTTL>
137     </roundRobin>
138 </connectionStrategy>
139 </appender>
140
141 <!-- 业务日志输出到LogStash -->
142 <appender name="LOG_STASH_BUSINESS"
143 class="net.logstash.logback.appender.LogstashTcpSocketAppender">
144     <destination>${LOG_STASH_HOST}:4562</destination>
145     <encoder charset="UTF-8"
146 class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">
147         <providers>
148             <timestamp>
149                 <timeZone>Asia/Shanghai</timeZone>
150             </timestamp>
151             <!-- 自定义日志输出格式 -->
152             <pattern>
153                 <pattern>
154                     {
155                         "project": "${PROJ_NAME:-}",
156                         "level": "%level",
157                         "service": "${APP_NAME:-}",
158                         "pid": "${PID:-}",
159                         "thread": "%thread",
160                         "class": "%logger",
161                         "message": "%message",
162                         "stack_trace": "%exception{20}"
163                     }
164                 </pattern>
165             </pattern>
166         </providers>
167     </encoder>
168 <!-- 当有多个LogStash服务时，设置访问策略为轮询 -->
169 <connectionStrategy>
170     <roundRobin>
171         <connectionTTL>5 minutes</connectionTTL>
172     </roundRobin>
173 </connectionStrategy>
174 </appender>
175
176 <!-- 接口访问记录日志输出到LogStash -->
177 <appender name="LOG_STASH_RECORD"
178 class="net.logstash.logback.appender.LogstashTcpSocketAppender">
179     <destination>${LOG_STASH_HOST}:4563</destination>
180     <encoder charset="UTF-8"
181 class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">
182         <providers>
183             <timestamp>
184                 <timeZone>Asia/Shanghai</timeZone>

```

```

181         </timestamp>
182         <!-- 自定义日志输出格式 -->
183         <pattern>
184             <pattern>
185                 {
186                 "project": "${PROJ_NAME:-}",
187                 "level": "%level",
188                 "service": "${APP_NAME:-}",
189                 "class": "%logger",
190                 "message": "%message"
191                 }
192             </pattern>
193         </pattern>
194     </providers>
195 </encoder>
196 <!-- 当有多个LogStash服务时，设置访问策略为轮询 -->
197 <connectionStrategy>
198     <roundRobin>
199         <connectionTTL>5 minutes</connectionTTL>
200     </roundRobin>
201 </connectionStrategy>
202 </appender>
203
204 <!-- root appender DEBUG级别以上日志 -->
205 <root level="DEBUG">
206     <!-- 控制台日志 -->
207     <appender-ref ref="CONSOLE"/>
208     <!-- 调试日志 -->
209     <appender-ref ref="LOG_STASH_DEBUG"/>
210     <!-- 错误日志 -->
211     <appender-ref ref="LOG_STASH_ERROR"/>
212 </root>
213 <!-- 业务日志 -->
214 <logger name="com.zeroone.star.projectlog" level="DEBUG">
215     <appender-ref ref="LOG_STASH_BUSINESS"/>
216 </logger>
217 <!-- 记录日志 -->
218 <logger name="com.zeroone.star.projectlog.component.WebLogAspect"
219 level="DEBUG">
220     <appender-ref ref="LOG_STASH_RECORD"/>
221 </logger>
222
223 <!--控制框架输出日志-->
224 <logger name="org.slf4j" level="INFO"/>
225 <logger name="springfox" level="INFO"/>
226 <logger name="io.swagger" level="INFO"/>
227 <logger name="org.springframework" level="INFO"/>
228 <logger name="org.hibernate.validator" level="INFO"/>
229 </configuration>

```

后面我们分开来对每个配置要点进行解释。

2 Logback配置

2.1 控制台日志配置

一般我们不需要自定义控制台输出，可以采用默认配置。

具体配置参考 `console-appender.xml`，该文件在 `spring-boot-${version}.jar` 下面。

```
1 <!-- 引用默认日志配置 -->
2 <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
3 <!-- 使用默认的控制台日志输出实现 -->
4 <include resource="org/springframework/boot/logging/logback/console-
  appender.xml"/>
```

2.2 springProperty

该标签可以从Spring Boot的配置文件中获取配置属性，比如说在不同环境下我们的 `Logstash` 服务地址是不一样的，我们就可以把该地址定义在 `application.yml` 来使用。

例如在 `application-dev.yml` 中定义了这些属性：

```
1 logstash:
2   host: localhost
```

在 `logback-spring.xml` 中就可以直接这样使用：

```
1 <!-- 应用名称 -->
2 <springProperty name="APP_NAME" scope="context"
3               source="spring.application.name" defaultValue="spring-boot"/>
4 <!-- Logstash访问host -->
5 <springProperty name="LOG_STASH_HOST" scope="context"
6               source="logstash.host" defaultValue="localhost"/>
```

2.3 filter

在 `Logback` 中有两种不同的过滤器，用来过滤日志输出。

ThresholdFilter：临界值过滤器，过滤掉低于指定临界值的日志，比如下面的配置将过滤掉所有低于 `INFO` 级别的日志。

```
1 <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
2   <level>INFO</level>
3 </filter>
```

LevelFilter：级别过滤器，根据日志级别进行过滤，比如下面的配置将过滤掉所有非 `ERROR` 级别的日志。

```
1 <filter class="ch.qos.logback.classic.filter.LevelFilter">
2   <level>ERROR</level>
3   <onMatch>ACCEPT</onMatch>
4   <onMismatch>DENY</onMismatch>
5 </filter>
```

2.4 appender

Appender可以用来控制日志的输出形式，主要有下面三种。

ConsoleAppender：控制日志输出到控制台的形式，比如在 `console-appender.xml` 中定义的默认控制台输出。

```
1 <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
2   <encoder>
3     <pattern>${CONSOLE_LOG_PATTERN}</pattern>
4   </encoder>
5 </appender>
```

RollingFileAppender：控制日志输出到文件的形式，可以控制日志文件生成策略，比如文件名称格式、超过多大重新生成文件以及删除超过多少天的文件。

```
1 <!-- ERROR日志输出到文件 -->
2 <appender name="FILE_ERROR"
3   class="ch.qos.logback.core.rolling.RollingFileAppender">
4   <rollingPolicy
5     class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
6     <!-- 设置文件命名格式 -->
7     <fileNamePattern>${LOG_FILE_PATH}/error/${APP_NAME}-%d{yyyy-MM-dd}-
8       %i.log</fileNamePattern>
9     <!-- 设置日志文件大小，超过就重新生成文件，默认10M -->
10    <maxFileSize>${LOG_FILE_MAX_SIZE:-10MB}</maxFileSize>
11    <!-- 日志文件保留天数，默认30天 -->
12    <maxHistory>${LOG_FILE_MAX_HISTORY:-30}</maxHistory>
13  </rollingPolicy>
14 </appender>
```

LogstashTcpSocketAppender：控制日志输出到 Logstash 的形式，可以用来配置 Logstash 的地址、访问策略以及日志的格式。

```
1 <!-- DEBUG日志输出到Logstash -->
2 <appender name="LOG_STASH_DEBUG"
3   class="net.logstash.logback.appender.LogstashTcpSocketAppender">
4   <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
5     <level>DEBUG</level>
6   </filter>
7   <destination>${LOG_STASH_HOST}:4560</destination>
8   <encoder charset="UTF-8"
9     class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">
10     <providers>
11       <timestamp>
12         <timeZone>Asia/Shanghai</timeZone>
13       </timestamp>
14       <!-- 自定义日志输出格式 -->
15       <pattern>
16         <pattern>
17           {
18             "project": "${PROJ_NAME:-}",
19             "level": "%level",
20             "service": "${APP_NAME:-}",
```

```

19         "pid": "${PID:-}",
20         "thread": "%thread",
21         "class": "%logger",
22         "message": "%message",
23         "stack_trace": "%exception{20}"
24     }
25     </pattern>
26 </pattern>
27 </providers>
28 </encoder>
29 <!-- 当有多个Logstash服务时，设置访问策略为轮询 -->
30 <connectionStrategy>
31     <roundRobin>
32         <connectionTTL>5 minutes</connectionTTL>
33     </roundRobin>
34 </connectionStrategy>
35 </appender>

```

2.5 logger

只有配置到logger节点上的appender才会被使用，logger用于配置哪种条件下的日志被打印，root是一种特殊的appender，下面介绍下日志划分的条件。

- 调试日志：所有的DEBUG级别以上日志；
- 错误日志：所有的ERROR级别日志；
- 业务日志：com.zeroone.star.projectlog 包下的所有DEBUG级别以上日志；
- 记录日志：com.zeroone.star.projectlog.component.WebLogAspect 类下所有DEBUG级别以上日志，该类是统计接口访问信息的 AOP 切面类。

还有一些使用框架内部的日志，DEBUG级别的日志对我们并没有啥用处，都可以设置为了INFO以上级别。

```

1 <!-- 控制框架输出日志 -->
2 <logger name="org.slf4j" level="INFO"/>
3 <logger name="springfox" level="INFO"/>
4 <logger name="io.swagger" level="INFO"/>
5 <logger name="org.springframework" level="INFO"/>
6 <logger name="org.hibernate.validator" level="INFO"/>

```

3 Logstash配置

接下来我们需要配置下 Logstash，让它可以分场景收集不同的日志，下面详细介绍下使用到的配置。

```

1 input {
2   tcp {
3     mode => "server"
4     host => "0.0.0.0"
5     port => 4560
6     codec => json_lines
7     type => "debug"
8   }
9   tcp {
10    mode => "server"
11    host => "0.0.0.0"

```



```

12     port => 4561
13     codec => json_lines
14     type => "error"
15 }
16 tcp {
17     mode => "server"
18     host => "0.0.0.0"
19     port => 4562
20     codec => json_lines
21     type => "business"
22 }
23 tcp {
24     mode => "server"
25     host => "0.0.0.0"
26     port => 4563
27     codec => json_lines
28     type => "record"
29 }
30 }
31 filter{
32     if [type] == "record" {
33         mutate {
34             remove_field => "port"
35             remove_field => "host"
36             remove_field => "@version"
37         }
38         json {
39             source => "message"
40             remove_field => ["message"]
41         }
42     }
43 }
44 output {
45     elasticsearch {
46         hosts => "es:9200"
47         action => "index"
48         codec => json
49         index => "project-%{type}-%{+YYYY.MM.dd}"
50         template_name => "project"
51     }
52 }

```

3.1 配置要点

- input: 使用不同端口收集不同类型的日志，从4560~4563开启四个端口；
- filter: 对于记录类型的日志，直接将JSON格式的message转化到source中去，便于搜索查看；
- output: 按类型、时间自定义索引格式。

4 Spring Boot配置

在Spring Boot中的配置可以直接用来覆盖Logback中的配置，比如logging.level.root就可以覆盖<root>节点中的level配置。

开发环境配置：application-dev.yml

```
1 logstash:
2   host: 192.168.220.127
3 logging:
4   level:
5     root: debug
```

测试环境配置: application-test.yml

```
1 logstash:
2   host: 192.168.220.128
3 logging:
4   level:
5     root: debug
```

生产环境配置: application-prod.yml

```
1 logstash:
2   host: 192.168.220.129
3 logging:
4   level:
5     root: info
```

5 Spring Boot集成案例

5.1 添加依赖

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5     https://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7
8   <groupId>com.zeroone.star</groupId>
9   <artifactId>project-log</artifactId>
10  <version>1.0.0-SNAPSHOT</version>
11
12  <properties>
13    <java.version>1.8</java.version>
14    <spring-boot.version>2.3.12.RELEASE</spring-boot.version>
15    <logstash-logback-encoder.version>6.6</logstash-logback-
16    encoder.version>
17    <knife4j.version>2.0.8</knife4j.version>
18    <hutool.version>5.8.3</hutool.version>
19    <mybatis.plus.version>3.4.3.4</mybatis.plus.version>
20  </properties>
21
22  <dependencies>
23    <dependency>
24      <groupId>org.springframework.boot</groupId>
25      <artifactId>spring-boot-starter-web</artifactId>
26    </dependency>
27    <dependency>
```

```
25         <groupId>org.springframework.boot</groupId>
26         <artifactId>spring-boot-starter-aop</artifactId>
27     </dependency>
28     <dependency>
29         <groupId>org.projectlombok</groupId>
30         <artifactId>lombok</artifactId>
31         <optional>true</optional>
32     </dependency>
33     <!-- logstash logback encoder -->
34     <dependency>
35         <groupId>net.logstash.logback</groupId>
36         <artifactId>logstash-logback-encoder</artifactId>
37     </dependency>
38     <!-- knife4j -->
39     <dependency>
40         <groupId>com.github.xiaoymin</groupId>
41         <artifactId>knife4j-spring-boot-starter</artifactId>
42     </dependency>
43     <!-- hutool -->
44     <dependency>
45         <groupId>cn.hutool</groupId>
46         <artifactId>hutool-all</artifactId>
47     </dependency>
48     <!-- mybatis-plus-extension -->
49     <dependency>
50         <groupId>com.baomidou</groupId>
51         <artifactId>mybatis-plus-extension</artifactId>
52     </dependency>
53 </dependencies>
54
55 <dependencyManagement>
56     <dependencies>
57         <!-- spring boot -->
58         <dependency>
59             <groupId>org.springframework.boot</groupId>
60             <artifactId>spring-boot-dependencies</artifactId>
61             <version>${spring-boot.version}</version>
62             <type>pom</type>
63             <scope>import</scope>
64         </dependency>
65         <!-- logstash logback encoder -->
66         <dependency>
67             <groupId>net.logstash.logback</groupId>
68             <artifactId>logstash-logback-encoder</artifactId>
69             <version>${logstash-logback-encoder.version}</version>
70         </dependency>
71         <!-- knife4j -->
72         <dependency>
73             <groupId>com.github.xiaoymin</groupId>
74             <artifactId>knife4j-spring-boot-starter</artifactId>
75             <version>${knife4j.version}</version>
76         </dependency>
77         <!-- hutool -->
78         <dependency>
79             <groupId>cn.hutool</groupId>
```

```

80         <artifactId>hutool-all</artifactId>
81         <version>${hutool.version}</version>
82         <optional>true</optional>
83     </dependency>
84     <!-- mybatis-plus-extension -->
85     <dependency>
86         <groupId>com.baomidou</groupId>
87         <artifactId>mybatis-plus-extension</artifactId>
88         <version>${mybaitis.plus.version}</version>
89     </dependency>
90 </dependencies>
91 </dependencyManagement>
92
93 <build>
94     <plugins>
95         <plugin>
96             <groupId>org.springframework.boot</groupId>
97             <artifactId>spring-boot-maven-plugin</artifactId>
98             <version>${spring-boot.version}</version>
99             <executions>
100                 <execution>
101                     <id>repackage</id>
102                     <goals>
103                         <goal>repackage</goal>
104                     </goals>
105                 </execution>
106             </executions>
107         </plugin>
108     </plugins>
109 </build>
110
111 </project>

```

5.2 添加 logback 配置

参考: [1分场景配置](#)

5.3 修改项目配置

下面是 application.yml 配置

```

1 application:
2   title: 测试项目
3 spring:
4   application:
5     name: PROJECT-LOG
6   profiles:
7     active: dev
8 server:
9   port: 8080

```

多环境配置, 参考[4 Spring Boot配置](#)

5.4 关键代码

5.4.1 访问记录对象

创建一个访问记录类，用于记录访问。

```
1  @Data
2  public class WebLog {
3      /**
4       * 操作描述
5       */
6      private String description;
7      /**
8       * 操作用户
9       */
10     private String username;
11     /**
12      * 操作时间
13      */
14     private Long startTime;
15     /**
16      * 消耗时间
17      */
18     private Integer spendTime;
19     /**
20      * 根路径
21      */
22     private String basePath;
23     /**
24      * URI
25      */
26     private String uri;
27     /**
28      * URL
29      */
30     private String url;
31     /**
32      * 请求类型
33      */
34     private String method;
35     /**
36      * IP地址
37      */
38     private String ip;
39     /**
40      * 请求参数
41      */
42     private Object parameter;
43     /**
44      * 请求返回的结果
45      */
46     private Object result;
47 }
```

5.4.2 AOP 切面

书写一个AOP切面拦截请求，生成请求记录日志

```
1  @Aspect
2  @Component
3  @Order(1)
4  public class WebLogAspect {
5      private static final Logger LOGGER =
6      LoggerFactory.getLogger(WebLogAspect.class);
7      @Pointcut("execution(public * com.zeroone.star.projectlog.controller.*.*
8      (..))")
9      public void webLog() {
10     }
11     @Around("webLog()")
12     public Object doAround(ProceedingJoinPoint joinPoint) throws Throwable {
13         long startTime = System.currentTimeMillis();
14         //获取当前请求对象
15         ServletRequestAttributes attributes = (ServletRequestAttributes)
16         RequestContextHolder.getRequestAttributes();
17         HttpServletRequest request =
18         Objects.requireNonNull(attributes).getRequest();
19         //记录请求信息
20         webLog webLog = new WebLog();
21         Object result = joinPoint.proceed();
22         Signature signature = joinPoint.getSignature();
23         MethodSignature methodSignature = (MethodSignature) signature;
24         Method method = methodSignature.getMethod();
25         if (method.isAnnotationPresent(ApiOperation.class)) {
26             ApiOperation apiOperation =
27             method.getAnnotation(ApiOperation.class);
28             webLog.setDescription(apiOperation.value());
29         }
30         long endTime = System.currentTimeMillis();
31         String urlStr = request.getRequestURL().toString();
32         webLog.setBasePath(StrUtil.removeSuffix(urlStr,
33         URLUtil.url(urlStr).getPath()));
34         webLog.setIp(request.getRemoteUser());
35         webLog.setMethod(request.getMethod());
36         webLog.setParameter(getParameter(method, joinPoint.getArgs()));
37         webLog.setResult(result);
38         webLog.setSpendTime((int) (endTime - startTime));
39         webLog.setStartTime(startTime);
40         webLog.setUri(request.getRequestURI());
41         webLog.setUrl(request.getRequestURL().toString());
42         Map<String, Object> logMap = new HashMap<>(5);
43         logMap.put("url", webLog.getUrl());
44         logMap.put("method", webLog.getMethod());
45         logMap.put("parameter", webLog.getParameter());
46         logMap.put("spendTime", webLog.getSpendTime());
47         logMap.put("description", webLog.getDescription());
48         LOGGER.info(Markers.appendEntries(logMap),
49         JSONUtil.parse(webLog.toString()));
50         return result;
51     }
52 }
```

```

45     /**
46      * 根据方法和传入的参数获取请求参数
47      */
48     private Object getParameter(Method method, Object[] args) {
49         List<Object> argList = new ArrayList<>();
50         Parameter[] parameters = method.getParameters();
51         for (int i = 0; i < parameters.length; i++) {
52             //将RequestBody注解修饰的参数作为请求参数
53             RequestBody requestBody =
parameters[i].getAnnotation(RequestBody.class);
54             if (requestBody != null) {
55                 argList.add(args[i]);
56             }
57             //将RequestParam注解修饰的参数作为请求参数
58             RequestParam requestParam =
parameters[i].getAnnotation(RequestParam.class);
59             String clsName = args[i].getClass().getName();
60             if (requestParam != null) {
61                 Map<String, Object> map = new HashMap<>(1);
62                 String key = parameters[i].getName();
63                 if (!StringUtils.isEmpty(requestParam.value())) {
64                     key = requestParam.value();
65                 }
66                 map.put(key, args[i]);
67                 argList.add(map);
68             }
69             //处理对象类型的RequestParam参数
70             else if (clsName.contains(".dto.") ||
clsName.contains(".query.")) {
71                 argList.add(args[i]);
72             }
73         }
74         if (argList.size() == 0) {
75             return null;
76         } else if (argList.size() == 1) {
77             return argList.get(0);
78         } else {
79             return argList;
80         }
81     }
82 }

```

5.4.3 Swagger配置

参考项目演示示例代码中，我们集成的是knife4j

5.4.4 其他领域模型

其他领域模型类参考示例源码

5.4.5 测试控制器

书写一个测试控制器

```
1  @Api(tags = "测试API")
2  @RestController
3  @RequestMapping("/test")
4  public class TestController {
5
6      @ApiOperation("获取所有")
7      @GetMapping("query-all")
8      public JsonVO<PageVO<SampleVO>> queryAll() {
9          List<SampleVO> voList = new ArrayList<>(10);
10         for (int i = 0; i < 10; i++) {
11             SampleVO sampleVO = new SampleVO();
12             sampleVO.setId(i + 1);
13             sampleVO.setAge(11);
14             sampleVO.setName("李四-" + (i + 1));
15             sampleVO.setSex("男");
16             voList.add(sampleVO);
17         }
18         PageVO<SampleVO> pv = new PageVO<>(1L, 12L, 10L, 1L, voList);
19         return JsonVO.success(pv);
20     }
21
22     @ApiOperation("添加数据")
23     @PostMapping("add")
24     public JsonVO<Long> add(SampleDTO data) {
25         return JsonVO.success(0L);
26     }
27
28     @ApiOperation("修改数据")
29     @PutMapping("modify")
30     public JsonVO<Long> modify(SampleDTO data) {
31         return JsonVO.success(1L);
32     }
33
34     @ApiOperation("删除数据")
35     @DeleteMapping("delete/{id}")
36     public JsonVO<Long> delete(@PathVariable("id") Long id) {
37         return JsonVO.success(id);
38     }
39
40     @ApiOperation("条件查询")
41     @GetMapping("query")
42     public JsonVO<PageVO<SampleVO>> query(SampleQuery query) {
43         SampleVO sampleVO = new SampleVO();
44         sampleVO.setId(1);
45         sampleVO.setAge(11);
46         sampleVO.setName(query.getName());
47         if ("张三".equals(query.getName())) {
48             sampleVO.setSex("男");
49         } else {
50             sampleVO.setSex("女");
51         }
52     }
53 }
```



```
52         List<SampleVO> voList = new ArrayList<>(1);
53         voList.add(sampleVO);
54         PageVO<SampleVO> pv = new PageVO<>(1L, 12L, 1L, 1L, voList);
55         return JsonVO.success(pv);
56     }
57 }
```

5.5 启动项目

启动项目，观察控制台是否有报错，如果没有报错，访问API文档页面，<http://localhost:8080/doc.html>

看到如下图所示的页面



host	localhost:8080	
basePath	/	
服务Url		
分组名称	default	
分组Url	/v2/api-docs	
分组location	/v2/api-docs	
接口统计信息	POST	1
	DELETE	1
	PUT	1
	GET	2

5.6 在Kibana中查看

访问你的 Kibana，没有修改端口的访问地址为，<http://ip:5601>

5.6.1 查看索引

D

Home

Recently viewed

Discover

Visualize

Dashboard

Canvas

Maps

Machine Learning

Metrics

Logs

APM

Uptime

SIEM

Dev Tools

Stack Monitoring

Management

Observability

APM

APM automatically collects in-depth performance metrics and errors from inside your applications.

Add APM

Logs

Ingest logs from popular sources and easily visualize in preconfigured dashboards.

Add log data

Add sample data

Load a data set and a Kibana dashboard

Upload

Import a dashboard

Visualize and Explore Data

APM

Automatically collect in-depth performance metrics and errors from inside your applications.

Canvas

Showcase your data in a pixel-perfect way.

进入管理菜单

点击，可以查看已经产生的索引

Elasticsearch

Index Management

Index Lifecycle Policies

Rollup Jobs

Transforms

Remote Clusters

Snapshot and Restore

License Management

8.0 Upgrade Assistant

Kibana

Index Patterns

Saved Objects

Spaces

Reporting

Advanced Settings

Index Management

Indices

Index Templates

Update your Elasticsearch indices individually or in bulk.

Search

<input type="checkbox"/>	Name	Health	Status
<input type="checkbox"/>	projectname-debug-2022.09.05	● green	open
<input type="checkbox"/>	projectname-business-2022.09.05	● green	open
<input type="checkbox"/>	projectname-record-2022.09.05	● green	open

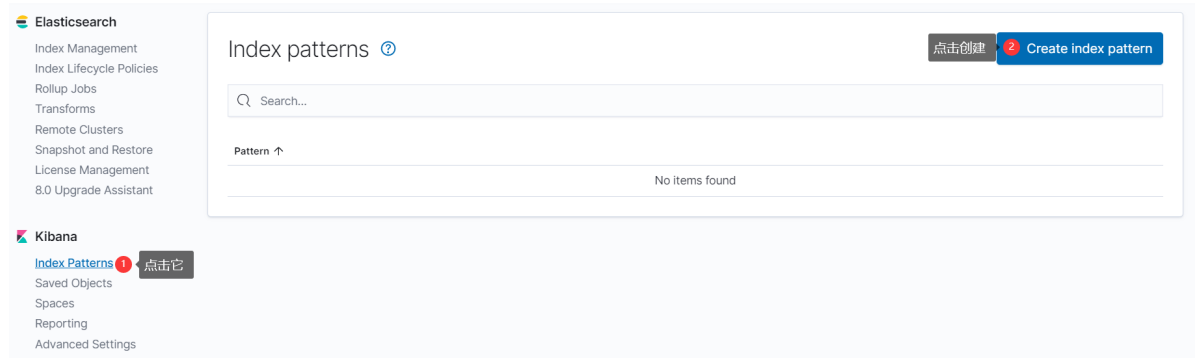
Rows per page: 10

点击进入索引管理

启动后创建的索引

5.6.2 索引模式

要使用 Kibana 查看索引详细信息，首先创建索引模式。



步骤1

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

☐ Include system indices

Step 1 of 2: Define index pattern

Index pattern

projectname-business-* 1

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

✓ **Success!** Your index pattern matches **1 index**.

projectname-business-2022.09.05

Rows per page: 10 ▾

> Next step 2

步骤2

Step 2 of 2: Configure settings

You've defined **projectname-business-*** as your index pattern. Now you can specify some settings before we create it.

Time Filter field name Refresh

@timestamp 1 ▾

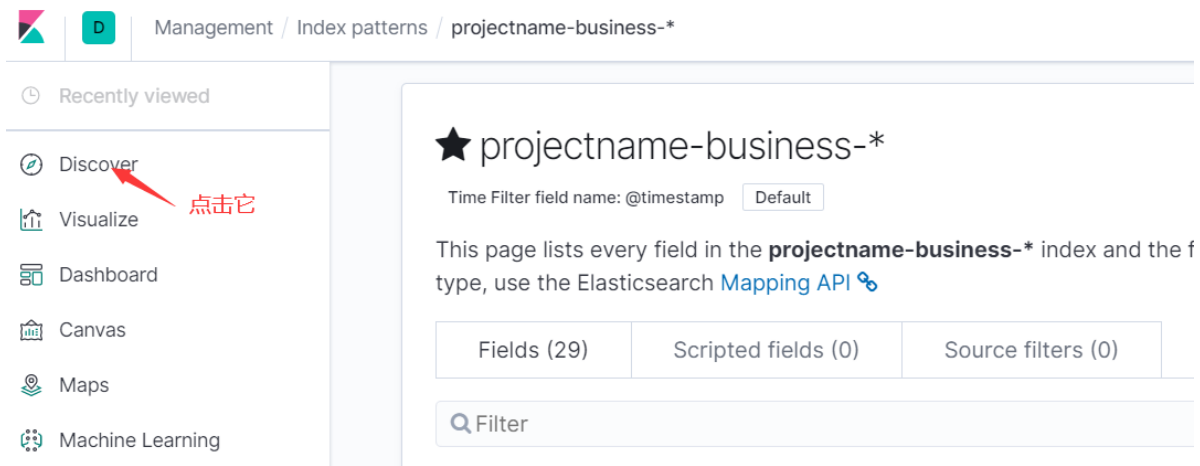
The Time Filter will use this field to filter your data by time.
You can choose not to have a time field, but you will not be able to narrow down your data by a time range.

> Show advanced options

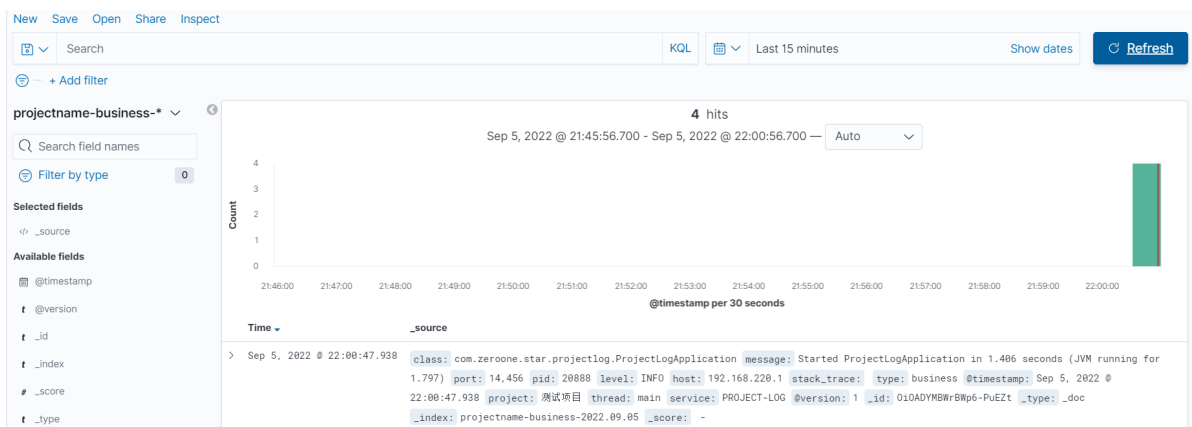
< Back 2 Create index pattern

5.6.3 查看日志

进入发现面板

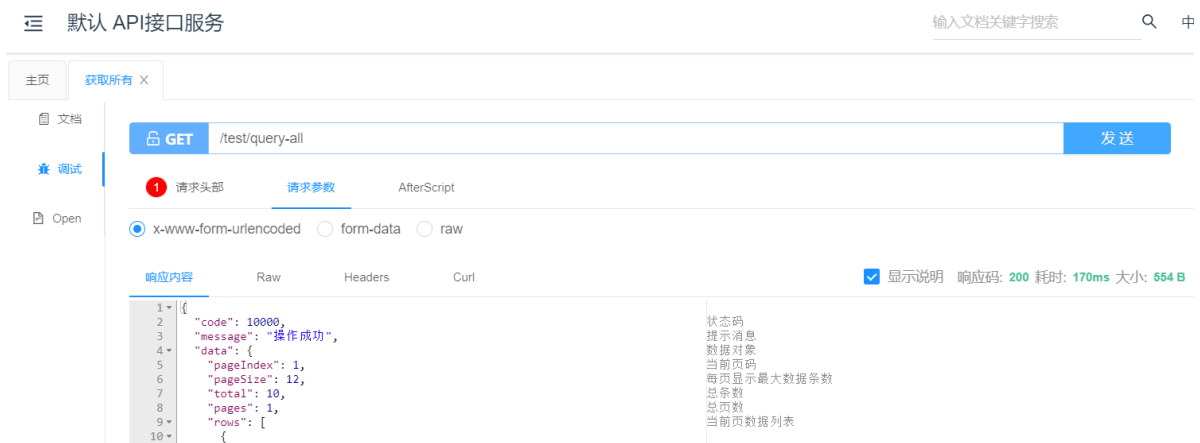


进入面板后可以看到已经有一些日志了，如下图所示



然后再API页面访问接口，查看日志生成情况

测试访问



查看测试访问记录日志，示例如下图所示

Step 5, 2022 @ 22:02:46.517

```
class: com.zeroone.star.projectlog.component.WebLogAspect message: {"description":"获取所有", "startTime":1662386566456, "spendTime":6, "basePath":"http://localhost:8080", "uri":"/test/query-all", "url":"http://localhost:8080/test/query-all", "method":"GET", "result":{"code":10000, "message":"操作成功", "data":{"pageIndex":1, "pageSize":12, "total":10, "pages":1, "rows":[{"id":1, "name":"李四-1", "sex":"男", "age":11}, {"id":2, "name":"李四-2", "sex":"男", "age":11}, {"id":3, "name":"李四-3", "sex":"男", "age":11}, {"id":4, "name":"李四-4", "sex":"男", "age":11}, {"id":5, "name":"李四-5", "sex":"男", "age":11}, {"id":6, "name":"李四-6", "sex":"男", "age":11}, {"id":7, "name":"李四-7", "sex":"男", "age":11}, {"id":8, "name":"李四-8", "sex":"男", "age":11}, {"id":9, "name":"李四-9", "sex":"男", "age":11}, {"id":10, "name":"李四-10", "sex":"男", "age":11}]}}
```

Expanded document

Table JSON

```
{  "_index": "projectname-business-2022.09.05",  "_type": "_doc",  "_id": "QCOCYMBWrBWP6-PrkZP",  "_version": 1,  "_score": null,  "_source": {    "class": "com.zeroone.star.projectlog.component.WebLogAspect",    "message": "{\n  \"description\": \"获取所有\",\n  \"startTime\": 1662386566456,\n  \"spendTime\": 6,\n  \"basePath\": \"http://localhost:8080\",\n  \"uri\": \"/test/query-all\",\n  \"url\": \"http://localhost:8080/test/query-all\",\n  \"method\": \"GET\",\n  \"result\": {\n    \"code\": 10000,\n    \"message\": \"操作成功\",\n    \"data\": {\n      \"pageIndex\": 1,\n      \"pageSize\": 12,\n      \"total\": 10,\n      \"pages\": 1,\n      \"rows\": [\n        {\n          \"id\": 1,\n          \"name\": \"李四-1\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 2,\n          \"name\": \"李四-2\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 3,\n          \"name\": \"李四-3\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 4,\n          \"name\": \"李四-4\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 5,\n          \"name\": \"李四-5\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 6,\n          \"name\": \"李四-6\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 7,\n          \"name\": \"李四-7\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 8,\n          \"name\": \"李四-8\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 9,\n          \"name\": \"李四-9\",\n          \"sex\": \"男\",\n          \"age\": 11\n        },\n        {\n          \"id\": 10,\n          \"name\": \"李四-10\",\n          \"sex\": \"男\",\n          \"age\": 11\n        }\n      ]\n    }\n  }\n}"
```

测试异常

```
@ApiOperation("获取所有")
@GetMapping("/query-all")
public JsonVO<PageVO<SampleVO>> queryAll() {
    //测试异常
    int b = 1 / 0;
    List<SampleVO> voList = new ArrayList<>(initialCapacity: 10);
    for (int i = 0; i < 10; i++) {
        SampleVO sampleVO = new SampleVO();
        sampleVO.setId(i + 1);
        sampleVO.setAge(11);
        sampleVO.setName("李四-" + (i + 1));
        sampleVO.setSex("男");
        voList.add(sampleVO);
    }
}
```

重启Spring Boot服务，然后访问一下query-all接口，然后在 kibana 中创建索引模式。

Step 1 of 2: Define index pattern

Index pattern

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

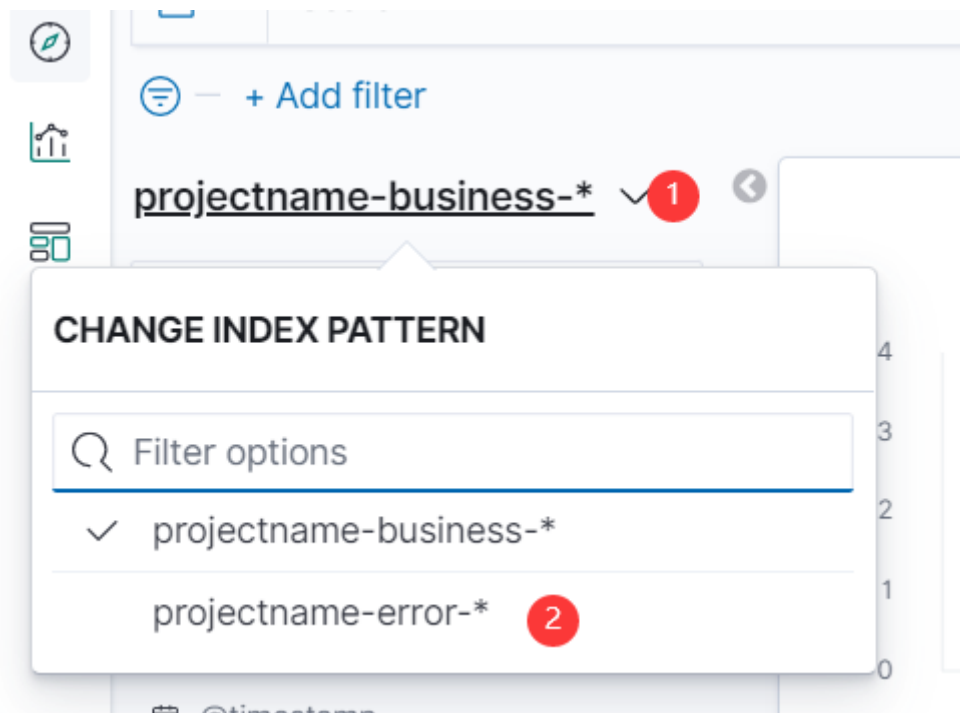
✓ Success! Your index pattern matches 1 index.

projectname-error-2022.09.05

Rows per page: 10

> Next step

在发现面板中查看错误日志



异常详情

Sep 5, 2022 @ 22:05:2 搜索

class: org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/].[/dispatcherServlet] message: Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed; nested exception is java.lang.ArithmeticException: / by zero] with root cause port: 1,225 pid: 17840 level: ERROR host: 192.168.220.1

stack_trace: java.lang.ArithmeticException: / by zero at com.zeroone.star.projectlog.controller.TestController.queryAll(TestController.java:30) at

Expanded document View surrounding documents View single document

Table JSON

```
{
  "_index": "projectname-error-2022.09.05",
  "_type": "_doc",
  "_id": "WioFDYMBWrBWp6-PRkaC",
  "_version": 1,
  "_score": null,
  "_source": {
    "class": "org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/].[/dispatcherServlet]",
    "message": "Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed; nested exception is java.lang.ArithmeticException: / by zero] with root cause",
    "port": 1225,
    "pid": "17840",
    "level": "ERROR",
    "host": "192.168.220.1",
  }
}
```

除0异常

stack_trace

```
java.lang.ArithmeticException: / by zero
    at com.zeroone.star.projectlog.controller.TestController.queryAll(TestController.java:30)
    at com.zeroone.star.projectlog.controller.TestController$$FastClassBySpringCGLIB$$def39c.invoke(<generated>)
    at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:218)
    at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:779)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163)
    at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750)
    at org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint.proceed(MethodInvocationProceedingJoinPoint.java:88)
    at com.zeroone.star.projectlog.component.WebLogAspect.doAround(WebLogAspect.java:54)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs(AbstractAspectJAdvice.java:644)
    at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod(AbstractAspectJAdvice.java:633)
    at org.springframework.aop.aspectj.AspectJAroundAdvice.invoke(AspectJAroundAdvice.java:70)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750)
    at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:95)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:750)
```