在分布式系统中，各种无关日志穿行其中，导致我们可能无法直接定位整个操作流程。因此，我们可能需要对一个用户的操作流程进行归类标记，比如使用线程+时间戳，或者用户身份标识等；如此，我们可以从大量日志信息中grep出某个用户的操作流程，或者某个时间的流转记录。

因此，这就有了 Slf4j MDC 方法。

MDC ( Mapped Diagnostic Contexts )

其目的是为了便于我们诊断线上问题而出现的方法工具类。虽然，Slf4j 是用来适配其他的日志具体实现包的，但是针对 MDC功能，目前只有logback 以及 log4j 支持

public class MDC {

//Put a context value as identified by key //into the current thread's context map.

public static void put(String key, String val);

//Get the context identified by the key parameter.

public static String get(String key);

//Remove the context identified by the key parameter.

public static void remove(String key);

//Clear all entries in the MDC.

public static void clear(); }

看看一个MDC使用的简单示例。

public class LogTest {

private static final Logger logger = LoggerFactory.getLogger(LogTest.class);

public static void main(String[] args) {

MDC.put("THREAD\_ID", String.valueOf(Thread.currentThread().getId()));

logger.info("纯字符串信息的info级别日志");

} }

然后看看logback的输出模板配置：

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

**<configuration>**

**<property** name="log.base" value="${catalina.base}/logs" **/>**

**<contextListener** class="ch.qos.logback.classic.jul.LevelChangePropagator"**>**

**<resetJUL>**true**</resetJUL>**

**</contextListener>**

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

**<encoder** charset="UTF-8"**>**

**<pattern>**[%d{yyyy-MM-dd HH:mm:ss} %highlight(%-5p) %logger.%M(%F:%L)] %X{THREAD\_ID} %msg%n**</pattern>**

**</encoder>**

**</appender>**

**<root** level="INFO"**>**

**<appender-ref** ref="console" **/>**

**</root>**

**</configuration>**

于是，就有了输出：

[2015-04-30 15:34:35 INFO io.github.ketao1989.log4j.LogTest.main(LogTest.java:29)] 1 纯字符串信息的info级别日志

在日志模板logback.xml 中，使用 %X{ }来占位，替换到对应的 MDC 中 key 的值。

refer: [http://www.ithao123.cn/content-8291525.html](%20http:/www.ithao123.cn/content-8291525.html)

logback.xml

{  
 "project": "zhihuishu-treenity",  
 "thread": "%thread",  
 "logger": "%logger",  
 "level": "%level",  
 "message": "%message"  
}

{

"timestamp": "2016-09-27 16:12:24.024",

"req.remoteHost": "192.168.70.109",

"req.method": "POST",

"req.userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0",

"req.principal": "shu",

"req.requestURI": "/course/createCourse",

"req.sessionId": "17EE1AFFB2F1BAEC7E7CDE26A9D1C8B1",

"project": "zhihuishu-treenity",

"thread": "http-apr-80-exec-8",

"logger": "com.zhihuishu.treenity.controller.HomeController",

"level": "WARN",

"message": "courseId:null"

},

{

"timestamp": "2016-09-27 16:07:28.028",

"req.remoteHost": "192.168.70.109",

"bq": "aspect",

"start": "1474963648461",

"method": "userCourseList",

"end": "1474963648475",

"request\_time": "14",

"lx": "dubbo",

"project": "zhihuishu-treenity",

"thread": "http-apr-80-exec-5",

"logger": "com.zhihuishu.treenity.service.course.CourseService",

"level": "INFO",

"message": "[\u5207\u9762\u65E5\u5FD7]\u63A5\u53E3\u6267\u884C\u6210\u529F\uFF0C\u8BF7\u6C42\u53C2\u6570\uFF1A687"

}

*<!-- 输出到文件 -->*<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">  
 <file>d:/data/rizhi\_log/yewu\_treenity.log</file>  
 <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">  
 <fileNamePattern>d:/data/rizhi\_log/yewu\_treenity.log.%d{yyyyMMdd}</fileNamePattern>  
 <maxHistory>60</maxHistory>  
 </rollingPolicy>  
 <encoder class="net.logstash.logback.encoder.LoggingEventCompositeJsonEncoder">  
 *<!-- 设置非立即刷新，内部使用BufferedOutputStream实现，默认缓冲区：8192bytes -->* <immediateFlush>true</immediateFlush>  
 <providers>  
 <timestamp>  
 <fieldName>timestamp</fieldName>  
 <pattern>yyyy-MM-dd HH:mm:ss.sss</pattern>  
 </timestamp>  
 <mdc />  
 <stackTrace>  
 <throwableConverter class="net.logstash.logback.stacktrace.ShortenedThrowableConverter">  
 <maxDepthPerThrowable>30</maxDepthPerThrowable>  
 <maxLength>2048</maxLength>  
 <shortenedClassNameLength>20</shortenedClassNameLength>  
 <exclude>sun\.reflect\..\*\.invoke.\*</exclude>  
 <exclude>net\.sf\.cglib\.proxy\.MethodProxy\.invoke</exclude>  
 <rootCauseFirst>true</rootCauseFirst>  
 </throwableConverter>  
 </stackTrace>  
 <pattern>  
 <pattern>  
 {  
 "project": "zhihuishu-treenity",  
 "thread": "%thread",  
 "logger": "%logger",  
 "level": "%level",  
 "message": "%message"  
 }  
 </pattern>  
 </pattern>  
 </providers>  
 </encoder>  
</appender>

*<!-- 输出到控制台 -->*<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
 <encoder charset="UTF-8">  
 <pattern>[zhihuishu-treenity] [%level] %d{yyyy-MM-dd HH:mm:ss.sss} %thread %logger %msg%n</pattern>  
 </encoder>  
</appender>

*<!-- 异步输出到文件 -->  
<!-- 异步日志Appender，内部使用FileAppender实现 -->*<appender name="ASYNC\_FILE" class="ch.qos.logback.classic.AsyncAppender">  
 <queueSize>1048576</queueSize>  
 <discardingThreshold>0</discardingThreshold>  
 <includeCallerData>false</includeCallerData>  
 <maxFlushTime>3000</maxFlushTime>  
 <appender-ref ref="FILE" />  
</appender>

*<!-- 默认输出到控制台 -->*<root level="WARN">  
 <appender-ref ref="STDOUT" />  
</root>  
  
*<!-- 业务日志输出到文件 -->*<logger name="com.zhihuishu.treenity" level="INFO" additivity="false">  
 <appender-ref ref="ASYNC\_FILE" />  
</logger>

appender下使用<includeCallerData>true</includeCallerData>

使得[%F:%L]行号输出生效

*<!-- 输出到控制台 -->*<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">  
 <includeCallerData>true</includeCallerData>  
 <encoder charset="UTF-8">  
 *<!--<pattern>[zhihuishu-course-interface] [%level] %d{yyyy-MM-dd HH:mm:ss.sss} %thread %logger %msg%n</pattern>-->* <pattern>%d{yyyy-MM-dd HH:mm:ss.sss} [%F:%L] - %msg%n</pattern>  
 </encoder>  
</appender>