芋道源码 —— 知识星球

我是一段不羁的公告！
记得给芳芳这 3 个项目加油，添加一个 STAR 噢。
https://github.com/YunaiV/SpringBoot-Labs
https://github.com/YunaiV/onemall
https://github.com/YunaiV/ruoyi-vue-pro
2019-06-05
Dubbo

精尽 Dubbo 源码解析 —— 日志适配

本文基于 Dubbo 2.6.1 版本，望知悉。

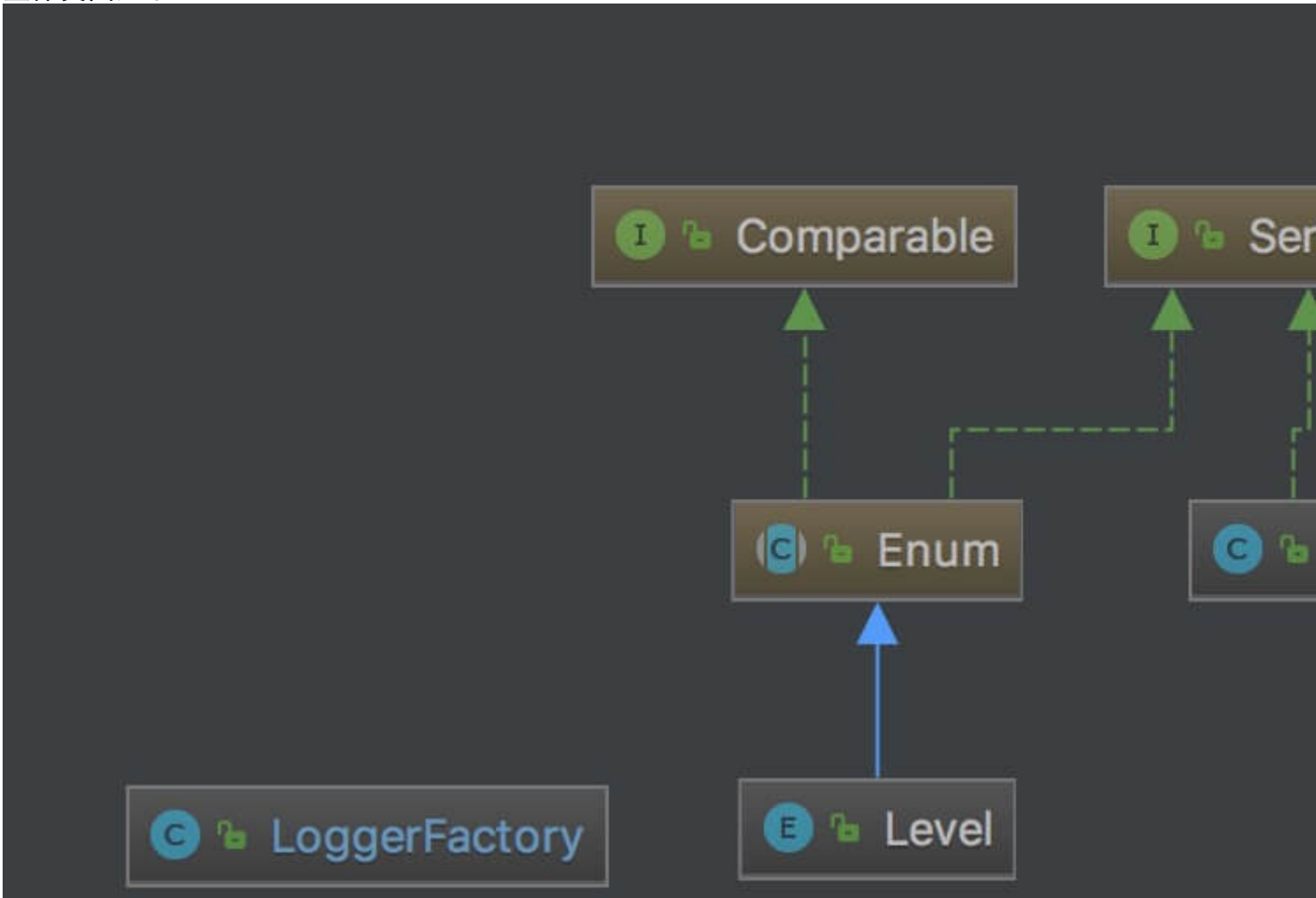
1. 概述

本文分享 Dubbo 的日志适配，对应文档为：

《Dubbo 用户指南 —— 日志适配》
《Dubbo 开发指南 —— 日志适配拓展》

自 2.2.1 开始，dubbo 开始内置 log4j、slf4j、jcl、jdk 这些日志框架的适配 1

整体类图如下：



2. LoggerFactory

com.alibaba.dubbo.common.logger.LoggerFactory ， 是 com.alibaba.dubbo.common.logger.Logger 的工厂类。

2.1 构造方法

```
/**
 * 已创建的 Logger 对应的映射
 *
 * key：类名
 */
private static final ConcurrentMap<String, FailsafeLogger> LOGGERS = new ConcurrentHashMap<String, FailsafeLogger>();
/**
 * 当前使用的 LoggerAdapter 日志适配器
 */
private static volatile LoggerAdapter LOGGER_ADAPTER;

private LoggerFactory() {}
```

通过设置的 LOGGER_ADAPTER ，创建对应的 com.alibaba.dubbo.common.logger.Logger 实现类，添加到 LOGGERS 中。下文，我们会详细解析。

2.2 setLoggerAdapter

#setLoggerAdapter(String loggerAdapter) 静态方法，设置 LoggerAdapter ，代码如下：

```
public static void setLoggerAdapter(String loggerAdapter) {
    if (loggerAdapter != null && loggerAdapter.length() > 0) {
        setLoggerAdapter(ExtensionLoader.getExtensionLoader(LoggerAdapter.class).getExtension(loggerAdapter));
    }
}
```

根据拓展名(注意，不是类名)，获得对应的 LoggerAdapter 实现类。并调用 #setLoggerAdapter(LoggerAdapter) 方法，设置 LoggerAdapter 对象。代码如下：

```
 1: public static void setLoggerAdapter(LoggerAdapter loggerAdapter) {
 2:     if (loggerAdapter != null) {
 3:         // 获得 Logger 对象，并打印日志，提示设置后的 LoggerAdapter 实现类
 4:         Logger logger = loggerAdapter.getLogger(LoggerFactory.class.getName());
 5:         logger.info("using logger: " + loggerAdapter.getClass().getName());
 6:         // 设置 LOGGER_ADAPTER 属性
 7:         LoggerFactory.LOGGER_ADAPTER = loggerAdapter;
 8:         // 循环，将原有已经生成的 LOGGER 缓存对象，全部重新生成替换
 9:         for (Map.Entry<String, FailsafeLogger> entry : LOGGERS.entrySet()) {
10:             entry.getValue().setLogger(LOGGER_ADAPTER.getLogger(entry.getKey()));
11:         }
12:     }
13: }
```

第 3 至 5 行：调用 LoggerAdapter#getLogger(String key) 方法，获得 LoggerFactory 的 对应

Logger 实现对象。然后，打印日志，提示设置后的 LoggerAdapter 实现类名。
LoggerAdapter 相关，在 「3. LoggerAdapter」 详细解析。
第 7 行：设置 LOGGER_ADAPTER 属性。
第 8 至 11 行：循环 LOGGERS，调用 LoggerAdapter#getLogger(String key) 方法，重新生成 LOGGER ，进行替换。

2.2.1 静态代码块

在 LoggerFactory 的静态代码块，会根据 "logger" 配置项，调用 #setLoggerAdapter(LoggerAdapter) 方法，进行设置 LOGGER_ADAPTER 属性。代码如下：

```
static {
    // 获得 "logger" 配置项
    String logger = System.getProperty("dubbo.application.logger");
    // 根据配置项，进行对应的 LoggerAdapter 对象
    if ("slf4j".equals(logger)) {
        setLoggerAdapter(new Slf4jLoggerAdapter());
    } else if ("jcl".equals(logger)) {
        setLoggerAdapter(new JclLoggerAdapter());
    } else if ("log4j".equals(logger)) {
        setLoggerAdapter(new Log4jLoggerAdapter());
    } else if ("jdk".equals(logger)) {
        setLoggerAdapter(new JdkLoggerAdapter());
    } else {
        // 未配置，按照 log4j > slf4j > apache common logger > jdk logger
        try {
            setLoggerAdapter(new Log4jLoggerAdapter());
        } catch (Throwable e1) {
            try {
                setLoggerAdapter(new Slf4jLoggerAdapter());
            } catch (Throwable e2) {
                try {
                    setLoggerAdapter(new JclLoggerAdapter());
                } catch (Throwable e3) {
                    setLoggerAdapter(new JdkLoggerAdapter());
                }
            }
        }
    }
}
```

代码易懂，看注释。
该方法适用于 LoggerFactory 未加载时，调用 System#setProperty("dubbo.application.logger", logger) 的初始化。

2.2.2 ApplicationConfig

ApplicationConfig 里，有 #setLogger(String logger) 方法，调用 #setLoggerAdapter(LoggerAdapter) 方法，进行设置 LOGGER_ADAPTER 属性。 代码如下：

```
private String logger;

public void setLogger(String logger) {
    this.logger = logger;
    // 设置 LoggerAdapter
```

```
        LoggerFactory.setLoggerAdapter(logger);
    }
```

通过如下方式配置时，都会调用该方法：

1. 命令行

```
java -Ddubbo.application.logger=log4j
```

2. 在 dubbo.properties 中指定

```
dubbo.application.logger=log4j
```

3. 在 dubbo.xml 中配置

```
<dubbo:application logger="log4j" />
```

2.3 getLogger

#getLogger(...) 方法，优先从 LOGGERS 中，获得对应的 Logger 对象。若不存在，则进行创建，并进行缓存到 LOGGERS 中。代码如下：

```
public static Logger getLogger(Class<?> key) {
    // 从缓存中，获得 Logger 对象
    FailsafeLogger logger = LOGGERS.get(key.getName());
    // 不存在，则进行创建，并进行缓存
    if (logger == null) {
        LOGGERS.putIfAbsent(key.getName(), new FailsafeLogger(LOGGER_ADAPTER.getLogger(key)));
        logger = LOGGERS.get(key.getName());
    }
    return logger;
}

public static Logger getLogger(String key) {
    // 从缓存中，获得 Logger 对象
    FailsafeLogger logger = LOGGERS.get(key);
    // 不存在，则进行创建，并进行缓存
    if (logger == null) {
        LOGGERS.putIfAbsent(key, new FailsafeLogger(LOGGER_ADAPTER.getLogger(key)));
        logger = LOGGERS.get(key);
    }
    return logger;
}
```

FailsafeLogger ，我们在 「4.1 FailsafeLogger」 中详细解析。

2.4 setLevel

#setLevel(Level level) 方法，设置日志级别。代码如下：

```
public static void setLevel(Level level) {
    LOGGER_ADAPTER.setLevel(level);
}
```

2.5 getLevel

#getLevel() 方法，获得日志级别。代码如下：

```
public static Level getLevel() {
    return LOGGER_ADAPTER.getLevel();
}
```

2.6 getFile

#getFile() 方法，获得当前日志文件。代码如下：

```
public static File getFile() {
    return LOGGER_ADAPTER.getFile();
}
```

3. LoggerAdapter

com.alibaba.dubbo.common.logger.LoggerAdapter ，Logger 适配器接口，负责对接不同日志库的 LoggerFactory 。接口方法如下：

```
Logger getLogger(Class<?> key);
Logger getLogger(String key);

Level getLevel();
void setLevel(Level level);

File getFile();
void setFile(File file);
```

3.1 Log4jLoggerAdapter

com.alibaba.dubbo.common.logger.log4j.Log4jLoggerAdapter ，实现 LoggerAdapter 接口，log4j 的 LoggerAdapter 实现类。

3.1.1 构造方法

```
/**
 * Root Logger 的文件，在构造方法中初始化
 */
private File file;
```

```
@SuppressWarnings("unchecked")
public Log4jLoggerAdapter() {
    try {
        // 获得 Root Logger 对象
        org.apache.log4j.Logger logger = LogManager.getRootLogger();
        if (logger != null) {
            // 循环每个 Logger 对象的 Appender 对象
            Enumeration<Appender> appenders = logger.getAllAppenders();
            if (appenders != null) {
                while (appenders.hasMoreElements()) {
                    // 当且仅当 FileAppender 时
                    Appender appender = appenders.nextElement();
                    if (appender instanceof FileAppender) {
                        FileAppender fileAppender = (FileAppender) appender;
                        String filename = fileAppender.getFile();
                        file = new File(filename);
                        break;
                    }
                }
            }
        }
    } catch (Throwable t) {
    }
}
```

`file` 属性，通过构造方法，进行初始化。

3.1.2 getLogger

```
@Override
public Logger getLogger(Class<?> key) {
    return new Log4jLogger(LogManager.getLogger(key));
}

@Override
public Logger getLogger(String key) {
    return new Log4jLogger(LogManager.getLogger(key));
}
```

先调用 `org.apache.log4j.LogManager#getLogger(...)` 方法，获得 `org.apache.log4j.Log` 对象。
再将 `org.apache.log4j.Log` 对象作为方法参数，创建 `com.alibaba.dubbo.common.logger.log4j.Log4jLogger` 对象。

3.1.3 getLevel

```
@Override
public Level getLevel() {
    return fromLog4jLevel(LogManager.getRootLogger().getLevel());
}
```

获得 Root Logger 的日志级别。

调用 `#fromLog4jLevel(Level)` ，将 Log4j 的日志级别转成 Dubbo 的日志级别。代码如下：

```
    private static Level fromLog4jLevel(org.apache.log4j.Level level) {
        if (level == org.apache.log4j.Level.ALL)
            return Level.ALL;
        if (level == org.apache.log4j.Level.TRACE)
            return Level.TRACE;
        if (level == org.apache.log4j.Level.DEBUG)
            return Level.DEBUG;
        if (level == org.apache.log4j.Level.INFO)
            return Level.INFO;
        if (level == org.apache.log4j.Level.WARN)
            return Level.WARN;
        if (level == org.apache.log4j.Level.ERROR)
            return Level.ERROR;
        // if (level == org.apache.log4j.Level.OFF)
        return Level.OFF;
    }
```

3.1.4 setLevel

```
@Override
public void setLevel(Level level) {
    LogManager.getRootLogger().setLevel(toLog4jLevel(level));
}
```

设置 Root Logger 的日志级别。

调用 `#toLog4jLevel(Level)` ，将 Dubbo 的日志级别转成 Log4j 的日志级别。代码如下：

```
    private static org.apache.log4j.Level toLog4jLevel(Level level) {
        if (level == Level.ALL)
            return org.apache.log4j.Level.ALL;
        if (level == Level.TRACE)
            return org.apache.log4j.Level.TRACE;
        if (level == Level.DEBUG)
            return org.apache.log4j.Level.DEBUG;
        if (level == Level.INFO)
            return org.apache.log4j.Level.INFO;
        if (level == Level.WARN)
            return org.apache.log4j.Level.WARN;
        if (level == Level.ERROR)
            return org.apache.log4j.Level.ERROR;
        // if (level == Level.OFF)
        return org.apache.log4j.Level.OFF;
    }
```

3.1.5 getFile

```
@Override
public File getFile() {
    return file;
}
```

3.1.6 setFile

```
@Override
public void setFile(File file) {
}
```

不支持设置。

3.2 JdkLoggerAdapter

类似 Log4jLoggerAdapter ，省略。

3.3 Slf4jLoggerAdapter

com.alibaba.dubbo.common.logger.log4j.Log4jLoggerAdapter ，实现 LoggerAdapter 接口，slf4j 的 LoggerAdapter 实现类。

SLF4J 不同于其他日志类库，与其它日志类库有很大的不同。SLF4J（Simple logging Facade for Java）不是一个真正的日志实现，而是一个抽象层（ abstraction layer），它允许你在后台使用任意一个日志类库。如果是在编写供内外部都可以使用的API或者通用类库，那么你真不会希望使用你类库的客户端必须使用你选择的日志类库。

因此，Slf4jLoggerAdapter 的实现方法中，操作 file 和 level 属性是无用。因为，具体的 file 和 level 由真正的日志实现决定。代码如下：

```
public class Slf4jLoggerAdapter implements LoggerAdapter {

    private Level level;
    private File file;

    @Override
    public Logger getLogger(String key) {
        return new Slf4jLogger(org.slf4j.LoggerFactory.getLogger(key));
    }

    @Override
    public Logger getLogger(Class<?> key) {
        return new Slf4jLogger(org.slf4j.LoggerFactory.getLogger(key));
    }

    @Override
    public Level getLevel() { // 无用
        return level;
    }

    @Override
    public void setLevel(Level level) { // 无用
        this.level = level;
    }

    @Override
    public File getFile() { // 无用
        return file;
```

```
        }

        @Override
        public void setFile(File file) { // 无用
            this.file = file;
        }

    }
```

- 也因此，打印日志的调用栈为：Dubbo Slf4jLogger => Slf4j Logger => 真正的 Logger 实现类。

3.4 JclLoggerAdapter

类似 Slf4jLoggerAdapter ，省略。

4. Logger

com.alibaba.dubbo.common.logger.Logger ，Logger 接口。代码如下：

```
void trace(String msg);
void trace(Throwable e);
void trace(String msg, Throwable e);
void debug(String msg);
void debug(Throwable e);
void debug(String msg, Throwable e);
void info(String msg);
void info(Throwable e);
void info(String msg, Throwable e);
void warn(String msg);
void warn(Throwable e);
void warn(String msg, Throwable e);
void error(String msg);
void error(Throwable e);
void error(String msg, Throwable e);

boolean isTraceEnabled();
boolean isDebugEnabled();
boolean isInfoEnabled();
boolean isWarnEnabled();
boolean isErrorEnabled();
```

方法的定义，复制自 Apache Common Logger 。

4.1 FailsafeLogger

com.alibaba.dubbo.common.logger.support.FailsafeLogger ，实现 Logger 接口，失败安全的 Logger 实现类。

我们以 #error(String msg) 实现方法，举例子。代码如下：

```
/**
 * Dubbo Logger 对象
 */
private Logger logger;

public FailsafeLogger(Logger logger) {
    this.logger = logger;
}


@Override
public void error(String msg) {
    try {
        logger.error(appendContextMessage(msg));
    } catch (Throwable t) {
    }
}
```

即使报错，也会被 try catch 掉。

#appendContextMessage(msg) 方法，拼接 Dubbo 的 version 和 host 到日志中。代码如下：

```
    private String appendContextMessage(String msg) {
        return " [DUBBO] " + msg + ", dubbo version: " + Version.getVersion() + ", current host: " + NetUtils.getLo
    }
```

其他实现方法，也类似该方法。

4.2 Log4jLogger

com.alibaba.dubbo.common.logger.log4j.Log4jLogger ，实现 Logger 接口，log4j 的 Logger 实现类。

我们以 #error(String msg) 实现方法，举例子。代码如下：

```
private static final String FQCN = FailsafeLogger.class.getName();

private final org.apache.log4j.Logger logger;

public Log4jLogger(org.apache.log4j.Logger logger) {
    this.logger = logger;
}

@Override
public void error(String msg) {
    logger.log(FQCN, Level.ERROR, msg, null);
}
```

每个实现方法，调用 Log4J Logger 对象，对应的方法。
其他实现方法，也类似该方法。

4.3 JdkLogger

类似 Log4jLogger ，省略。

4.4 Slf4jLogger

类似 Log4jLogger ，省略。

4.5 JclLogger

类似 Log4jLogger ，省略。

5. Level

com.alibaba.dubbo.common.logger.Level ， 日志级别枚举。代码如下：

```
public enum Level {

    ALL,
    TRACE,
    DEBUG,
    INFO,
    WARN,
    ERROR,
    OFF

}
```

666. 彩蛋

一般情况下，我们使用 slf4j ，并使用 slf4j 对应的适配库。具体可参考：《dubbo应用配置logback日志》 。

文章目录

1. 1. 概述
2. 2. LoggerFactory
    1. 2.1. 2.1 构造方法
    2. 2.2. 2.2 setLoggerAdapter
        1. 2.2.1. 2.2.1 静态代码块
        2. 2.2.2. 2.2.2 ApplicationConfig
    3. 2.3. 2.3 getLogger
    4. 2.4. 2.4 setLevel
    5. 2.5. 2.5 getLevel
    6. 2.6. 2.6 getFile
3. 3. LoggerAdapter
    1. 3.1. 3.1 Log4jLoggerAdapter
        1. 3.1.1. 3.1.1 构造方法
        2. 3.1.2. 3.1.2 getLogger
        3. 3.1.3. 3.1.3 getLevel
        4. 3.1.4. 3.1.4 setLevel
        5. 3.1.5. 3.1.5 getFile
        6. 3.1.6. 3.1.6 setFile
    2. 3.2. 3.2 JdkLoggerAdapter
    3. 3.3. 3.3 Slf4jLoggerAdapter
    4. 3.4. 3.4 JclLoggerAdapter
4. 4. Logger
    1. 4.1. 4.1 FailsafeLogger
    2. 4.2. 4.2 Log4jLogger
    3. 4.3. 4.3 JdkLogger
    4. 4.4. 4.4 Slf4jLogger
    5. 4.5. 4.5 JclLogger
5. 5. Level
6. 6. 666. 彩蛋

回到首页