crazyhzm 265

Dubbo源码解析（十二）远程通信——Telnet

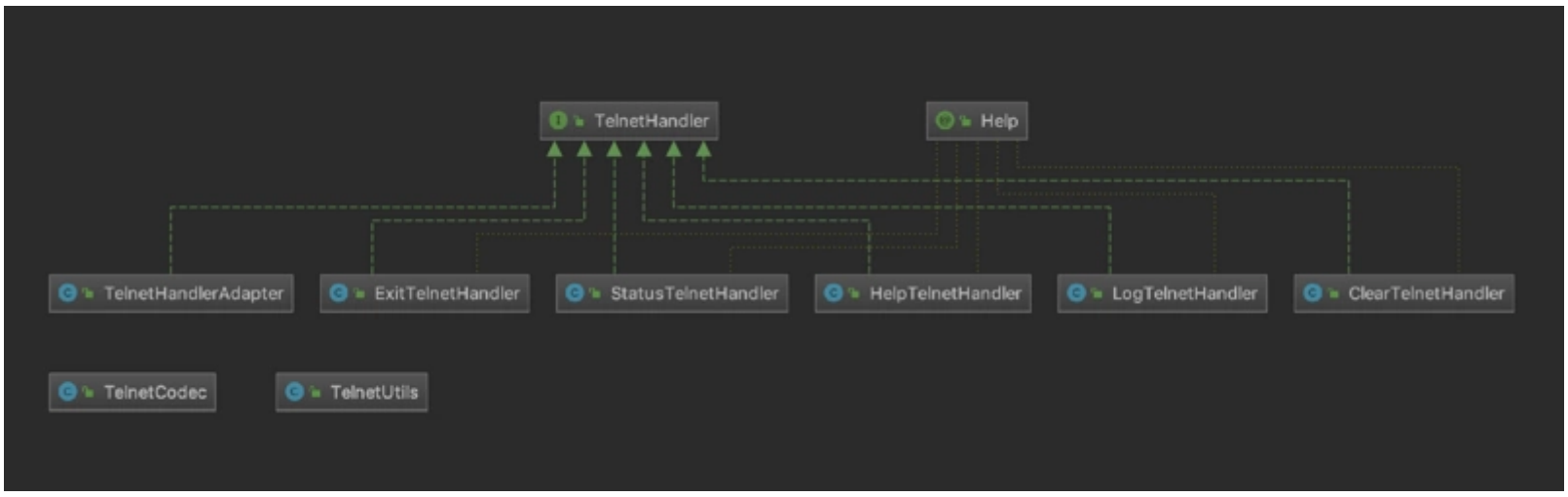
java dubbo telnet 阅读约 45 分钟

远程通讯——Telnet

> 目标：介绍telnet的相关实现逻辑、介绍dubbo-remoting-api中的telnet包内的源码解析。

前言

从dubbo 2.0.5开始，dubbo开始支持通过 telnet 命令来进行服务治理。本文就是讲解一些公用的telnet命令的实现。下面来看一下telnet实现的类图：



可以看到，实现了TelnetHandler接口的有六个类，除了TelnetHandlerAdapter是以外，其他五个分别对应了clear、exit、help、log、status命令的实现，具体用来干嘛，请看官方文档的介绍。

源码分析

（一）TelnetHandler

```
@SPI
public interface TelnetHandler {

    /**
     * telnet.
     * 处理对应的telnet命令
     * @param channel
     * @param message telnet命令
     */
    String telnet(Channel channel, String message) throws RemotingException;

}
```

该接口上telnet命令处理器接口，是一个可扩展接口。它定义了一个方法，就是处理相关的telnet命令。

（二）TelnetHandlerAdapter

该类继承了ChannelHandlerAdapter，实现了TelnetHandler接口，是TelnetHandler的适配器类，负责在接收到HeaderExchangeHandler发来的telnet命令后分发给对应的TelnetHandler实现类去实现，并且返回命令结果。

```
public class TelnetHandlerAdapter extends ChannelHandlerAdapter implements TelnetHandler {

    /**
     * 扩展加载器
     */
    private final ExtensionLoader<TelnetHandler> extensionLoader =
ExtensionLoader.getExtensionLoader(TelnetHandler.class);

    @Override
    public String telnet(Channel channel, String message) throws RemotingException {
        // 获得提示键配置，用于nc获取信息时不显示提示符
        String prompt = channel.getUrl().getParameterAndDecoded(Constants.PROMPT_KEY, Constants.DEFAULT_PROMPT);
        boolean noprompt = message.contains("--no-prompt");
        message = message.replace("--no-prompt", "");
        StringBuilder buf = new StringBuilder();
        // 删除头尾空白符的字符串
        message = message.trim();
        String command;
        // 获得命令
        if (message.length() > 0) {
            int i = message.indexOf(' ');
            if (i > 0) {
                // 获得命令
                command = message.substring(0, i).trim();
                // 获得参数
                message = message.substring(i + 1).trim();
```

该类只实现了telnet方法，其中的逻辑还是比较清晰，就是根据对应的命令去让对应的实现类产生命令结果。

（三）ClearTelnetHandler

该类实现了TelnetHandler接口，封装了clear命令的实现。

```
@Activate
@Help(parameter = "[lines]", summary = "Clear screen.", detail = "Clear screen.")
public class ClearTelnetHandler implements TelnetHandler {

    @Override
    public String telnet(Channel channel, String message) {
        // 清除屏幕上的内容行数
        int lines = 100;
        if (message.length() > 0) {
            // 如果不是一个数字
            if (!StringUtils.isInteger(message)) {
                return "Illegal lines " + message + ", must be integer.";
            }
            lines = Integer.parseInt(message);
        }
        StringBuilder buf = new StringBuilder();
        // 一行一行清除
        for (int i = 0; i < lines; i++) {
            buf.append("\r\n");
        }
        return buf.toString();
    }

}
```

（四）ExitTelnetHandler

该类实现了TelnetHandler接口，封装了exit命令的实现。

```
@Activate
@Help(parameter = "", summary = "Exit the telnet.", detail = "Exit the telnet.")
public class ExitTelnetHandler implements TelnetHandler {

    @Override
    public String telnet(Channel channel, String message) {
        // 关闭通道
        channel.close();
        return null;
    }

}
```

(五) HelpTelnetHandler

该类实现了TelnetHandler接口，封装了help命令的实现。

```
@Activate
@Help(parameter = "[command]", summary = "Show help.", detail = "Show help.")
public class HelpTelnetHandler implements TelnetHandler {

    /**
     * 扩展加载器
     */
    private final ExtensionLoader<TelnetHandler> extensionLoader =
ExtensionLoader.getExtensionLoader(TelnetHandler.class);

    @Override
    public String telnet(Channel channel, String message) {
        // 如果需要查看某一个命令的帮助
        if (message.length() > 0) {
            if (!extensionLoader.hasExtension(message)) {
                return "No such command " + message;
            }
            // 获得对应的扩展实现类
            TelnetHandler handler = extensionLoader.getExtension(message);
            Help help = handler.getClass().getAnnotation(Help.class);
            StringBuilder buf = new StringBuilder();
            // 生成命令和帮助信息
            buf.append("Command:\r\n    ");
            buf.append(message + " " + help.parameter().replace("\r\n", " ").replace("\n", " "));
            buf.append("\r\nSummary:\r\n    ");
            buf.append(help.summary().replace("\r\n", " ").replace("\n", " "));
```

help分为了需要查看某一个命令的帮助还是查看全部命令的帮助。

(六) LogTelnetHandler

该类实现了TelnetHandler接口，封装了log命令的实现。

```
@Activate
@Help(parameter = "level", summary = "Change log level or show log ", detail = "Change log level or show log")
public class LogTelnetHandler implements TelnetHandler {

    public static final String SERVICE_KEY = "telnet.log";

    @Override
    public String telnet(Channel channel, String message) {
        long size = 0;
        File file = LoggerFactory.getFile();
        StringBuffer buf = new StringBuffer();
        if (message == null || message.trim().length() == 0) {
            buf.append("EXAMPLE: log error / log 100");
        } else {
            String str[] = message.split(" ");
            if (!StringUtils.isInteger(str[0])) {
                // 设置日志级别
                LoggerFactory.setLevel(Level.valueOf(message.toUpperCase()));
            } else {
                // 获得日志长度
                int SHOW_LOG_LENGTH = Integer.parseInt(str[0]);

                if (file != null && file.exists()) {
                    try {
                        FileInputStream fis = new FileInputStream(file);
                        try {
```

log命令实现原理就是从日志文件中把日志信息读取出来。

（七）StatusTelnetHandler

该类实现了TelnetHandler接口，封装了status命令的实现。

```
@Activate
@Help(parameter = "[-l]", summary = "Show status.", detail = "Show status.")
public class StatusTelnetHandler implements TelnetHandler {

    private final ExtensionLoader<StatusChecker> extensionLoader =
ExtensionLoader.getExtensionLoader(StatusChecker.class);

    @Override
    public String telnet(Channel channel, String message) {
        // 显示状态列表
        if (message.equals("-l")) {
            List<StatusChecker> checkers = extensionLoader.getActivateExtension(channel.getUrl(), "status");
            String[] header = new String[]{"resource", "status", "message"};
            List<List<String>> table = new ArrayList<List<String>>();
            Map<String, Status> statuses = new HashMap<String, Status>();
            if (checkers != null && !checkers.isEmpty()) {
                // 遍历各个资源的状态，如果一个当全部 OK 时则显示 OK，只要有一个 ERROR 则显示 ERROR，只要有一个 WARN 则
显示 WARN
                for (StatusChecker checker : checkers) {
                    String name = extensionLoader.getExtensionName(checker);
                    Status stat;
                    try {
                        stat = checker.check();
                    } catch (Throwable t) {
                        stat = new Status(Status.Level.ERROR, t.getMessage());
                    }
```

（八）Help

该接口是帮助文档接口

```
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE})
public @interface Help {

    String parameter() default "";

    String summary();

    String detail() default "";

}
```

可以看上在每个命令的实现类上都加上了@Help注解，为了添加一些帮助文案。

（九）TelnetUtils

该类是Telnet命令的工具类，其中逻辑我就不介绍了。

（十）TelnetCodec

该类继承了TransportCodec，是telnet的编解码类。

1.属性

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

/**
 * 历史命令列表
 */
private static final String HISTORY_LIST_KEY = "telnet.history.list";

/**
 * 历史命令位置，就是用上下键来找历史命令
 */
private static final String HISTORY_INDEX_KEY = "telnet.history.index";

/**
 * 向上键
 */
private static final byte[] UP = new byte[]{27, 91, 65};

/**
 * 向下键
 */
private static final byte[] DOWN = new byte[]{27, 91, 66};

/**
 * 回车
 */
private static final List<?> ENTER = Arrays.asList(new Object[]{new byte[]{'\r', '\n'} /* Windows Enter */, new
```

2.getCharset

```
private static Charset getCharset(Channel channel) {
    if (channel != null) {
        // 获得属性设置
        Object attribute = channel.getAttribute(Constants.CHARSET_KEY);
        // 返回指定字符集的charset对象。
        if (attribute instanceof String) {
            try {
                return Charset.forName((String) attribute);
            } catch (Throwable t) {
                logger.warn(t.getMessage(), t);
            }
        } else if (attribute instanceof Charset) {
            return (Charset) attribute;
        }
        URL url = channel.getUrl();
        if (url != null) {
            String parameter = url.getParameter(Constants.CHARSET_KEY);
            if (parameter != null && parameter.length() > 0) {
                try {
                    return Charset.forName(parameter);
                } catch (Throwable t) {
                    logger.warn(t.getMessage(), t);
                }
            }
        }
    }
```

该方法是获得通道的字符集，根据url中编码来获得字符集，默认是utf-8。

3.encode

```
@Override
public void encode(Channel channel, ChannelBuffer buffer, Object message) throws IOException {
    // 如果需要编码的是 telnet 命令结果
    if (message instanceof String) {
        //如果为客户端侧的通道message直接返回
        if (isClientSide(channel)) {
            message = message + "\r\n";
        }
        // 获得字节数组
        byte[] msgData = ((String) message).getBytes(getCharset(channel).name());
        // 写入缓冲区
        buffer.writeBytes(msgData);
    } else {
        super.encode(channel, buffer, message);
    }
}
```

该方法是编码方法。

4.decode

```java
@Override
public Object decode(Channel channel, ChannelBuffer buffer) throws IOException {
    // 获得缓冲区可读的字节
    int readable = buffer.readableBytes();
    byte[] message = new byte[readable];
    // 从缓冲区读数据
    buffer.readBytes(message);
    return decode(channel, buffer, readable, message);
}

@SuppressWarnings("unchecked")
protected Object decode(Channel channel, ChannelBuffer buffer, int readable, byte[] message) throws IOException {
    // 如果是客户端侧，直接返回结果
    if (isClientSide(channel)) {
        return toString(message, getCharset(channel));
    }
    // 检验消息长度
    checkPayload(channel, readable);
    if (message == null || message.length == 0) {
        return DecodeResult.NEED_MORE_INPUT;
    }

    // 如果回退
    if (message[message.length - 1] == '\b') { // Windows backspace echo
        try {
            boolean doublechar = message.length >= 3 && message[message.length - 3] < 0; // double byte char
```

该方法是编码。

后记

> 该部分相关的源码解析地址：https://github.com/CrazyHZM/i...

该文章讲解了telnet的相关实现逻辑,本文有兴趣的朋友可以看看。下一篇我会讲解基于grizzly实现远程通信部分。

crazyhzm
265

关注作者

0 条评论

得票 • 时间

撰写评论 ...

提交评论

推荐阅读

dubbo源码解析（一）Hello,Dubbo

你好，dubbo，初次见面，我想和你交个朋友。先给出一套官方的说法：ApacheDubbo是一款高性能、轻量级基于Java的RPC开源...

CrazyHzm • 阅读 633