Flink数据源拆解分析(WikipediaEditsSource)

Original 程序员欣宸 程序员欣宸 Yesterday

Wikipedia Edit Stream是Flink官网上的经典demo，功能是实时处理来自维基百科的消息，消息的内容是当前每个用户对维基内容的操作，地址是：
https://ci.apache.org/projects/flink/flink-docs-release-1.2/quickstart/run_example_quickstart.html

在demo中，WikipediaEditsSource类作为数据源负责向Flink提供实时消息，今天咱们一起来分析其源码，了解Flink是怎么获取到来自Wiki的实时数据的，这对我们今后做自定义数据源也有很好的参考作用；

官方解释

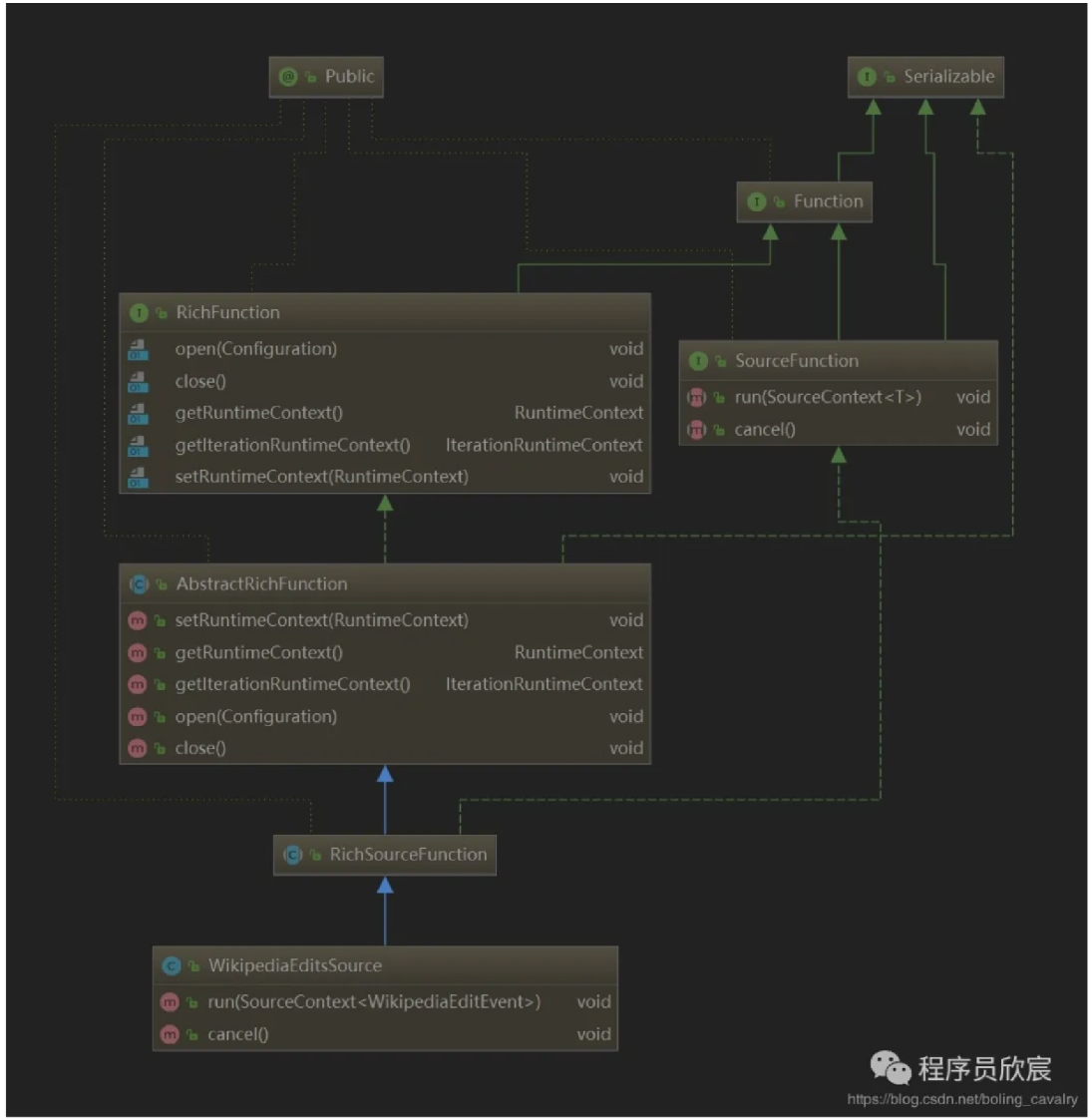
以下是官网对消息来源的说明，维基百科提供了一个IRC协议的通道，从这个通道可以获取对维基百科所做的编辑行为的日志：

```
Wikipedia provides an IRC channel where all edits to the wiki are logged.
```

IRC是应用层协议，更多细节请看：https://en.wikipedia.org/wiki/Internet_Relay_Chat

继承关系

先看WikipediaEditsSource类的继承关系，做个初步了解，如下图：



如上图所示，RichFunction接口负责资源开启关闭以及环境上下文，而SourceFunction接口则是和数据生产行为的开始和停止有关，这些接口最终都在WikipediaEditSource实现；

构造方法

通过构造方法来了解有哪些参数被确定了：

```
        //远程连接的域名
        public static final String DEFAULT_HOST = "irc.wikimedia.org";
        //远程连接的端口
        public static final int DEFAULT_PORT = 6667;
        //IRC协议的channel
        public static final String DEFAULT_CHANNEL = "#en.wikipedia";

        private final String host;
        private final int port;
        private final String channel;

        public WikipediaEditsSource() {
                this(DEFAULT_HOST, DEFAULT_PORT, DEFAULT_CHANNEL);
        }

        public WikipediaEditsSource(String host, int port, String channel) {
                this.host = host;
                this.port = port;
                this.channel = Objects.requireNonNull(channel);
        }
```

通过上述代码可以见到，数据的来源是irc.wikimedia.org这个网址；

主业务代码

主要的业务逻辑是WikipediaEditsSource的run方法，该方法在任务启动的时候会被StreamSource.run方法调用：

```
        @Override
        public void run(SourceContext<WikipediaEditEvent> ctx) throws Exception {
                try (WikipediaEditEventIrcStream ircStream = new WikipediaEditEventIrcStream(host, port)) {
                        // 创建一个IRC协议的连接
                        ircStream.connect();
                        //进入指定的channel
```

```
                        ircStream.join(channel);

                        try {
                                while (isRunning) {
                                        //从阻塞队列中获取数据
                                        WikipediaEditEvent edit = ircStream.getEdits().poll(100, TimeUnit.MILLISECONDS);
                                        //如果取到了数据，就调用ctx.collect方法，将数据生产到Flink环境，给其他operator使用
                                        if (edit != null) {
                                                ctx.collect(edit);
                                        }
                                }
                        } finally {
                                //结束时要向服务器发送数据表示离开
                                ircStream.leave(channel);
                        }
                }
        }
```

上面的代码，我们挑几处重要的展开看一看；

和维基百科消息服务器建立连接后做的事情

1. 为了弄明白Flink是如何与维基百科的数据源建立连接的，先把ircStream.connect()这段代码展开，对应的是IRCConnection类的connect方法：

```
public void connect() throws IOException {
                if (level != 0) // otherwise disconnected or connect
                        throw new SocketException("Socket closed or already open ("+ level +")");
                IOException exception = null;
                Socket s = null;
                for (int i = 0; i < ports.length && s == null; i++) {
                        try {
                                //建立的是普通Socket连接
                                s = new Socket(host, ports[i]);
                                exception = null;
```

```
                        } catch (IOException exc) {
                                if (s != null)
                                        s.close();
                                s = null;
                                exception = exc;
                        }
                }
                if (exception != null)
                        throw exception; // connection wasn't successful at any port

                prepare(s);
        }
```

上述代码表明，Flink与维基百科的数据源服务器之间建立的是普通的Socket连接，至于IRC协议，都是在这个Socket连接的通道里的一些读写操作；

2. 上面的prepare方法比较关键，展开看看：

```
protected void prepare(Socket s) throws IOException {
                if (s == null)
                        throw new SocketException("Socket s is null, not connected");
                socket = s;
                level = 1;
                s.setSoTimeout(timeout);
                in  = new BufferedReader(new InputStreamReader(s.getInputStream(),
                                encoding));
                out = new PrintWriter(new OutputStreamWriter(s.getOutputStream(),
                                encoding));

                //IRCConnection是Thread的子类，执行start方法就表明会启动一个线程来执行IRCConnection的run方法
                start();
                //遵守IRC协议约定，发送一些注册相关的内容
                register();
        }
```

可以看出，prepare方法做了两个重要的事情：启动一个子线程、发送IRC协议的注册信息，接下来看启动的子线程做了什么；

3. 打开IRCConnection的run方法：

```
public void run() {
                try {
                        String line;
                        while (!isInterrupted()) {
                                line = in.readLine();
                                if (line != null)
                                        get(line);
                                else
                                        close();
                        }
                } catch (IOException exc) {
                        close();
                }
        }
```

run方法中的内容很简单，就是让这个子线程负责读取远端发送的字符串，每读到一行就调用get方法去处理；

4. get方法的内容很多，做的事情是根据IRC协议解析这个字符串再做不同的处理，这里我们只要关注下面这段，就是收到一条业务消息后如何处理：

```
//每当有人编辑了维基百科，这里就会收到一条command为PRIVMSG的记录
if (command.equalsIgnoreCase("PRIVMSG")) { // MESSAGE
                        IRCUser user = p.getUser();
                        String middle = p.getMiddle();
                        String trailing = p.getTrailing();
                        for (int i = listeners.length - 1; i >= 0; i--)
                                //调用listener的onPrivmsg方法
                                listeners[i].onPrivmsg(middle, user, trailing);
                }
```

如上所示，每收到一条远端发来的消息，都会调用listener的onPrivmsg方法，这里的注册的linstener是WikipediaIrcChannelListener对象；

5. 打开WikipediaIrcChannelListener的onPrivmsg方法，看看收到消息后做了什么：

```
@Override
public void onPrivmsg(String target, IRCUser user, String msg) {
        LOG.debug("[{}] {}: {}.", target, user.getNick(), msg);
        //根据消息构造一个WikipediaEditEvent对象，就是Flink的业务流程中用到的数据对象
        WikipediaEditEvent event = WikipediaEditEvent.fromRawEvent(
                System.currentTimeMillis(),
                target,
                msg);

        if (event != null) {
                //eidts是个阻塞队列，WikipediaEditEvent被放入队列
                if (!edits.offer(event)) {
                        LOG.debug("Dropping message, because of full queue.");
                }
        }
}
```

上面的代码已经分析把主要逻辑展现出来了，从Socket读到的数据被解析成Flink实时计算时用到的WikipediaEditEvent对象后，被放入阻塞队列中，这也就是负责读取的子线程的主要工作了；

如何消费队列中的数据

前面的分析中我们得知：收到的数据被放入了阻塞队列中，现在回到WikipediaEditsSource的run方法再看看，这里面就有从阻塞队列取出数据的操作：

```
while (isRunning) {
                //从阻塞队列中获取数据
                WikipediaEditEvent edit = ircStream.getEdits().poll(100, TimeUnit.MILLISECONDS);
                //如果取到了数据，就调用ctx.collect方法，将数据生产到Flink环境，给其他operator使用
                if (edit != null) {
                        ctx.collect(edit);
                }
```

```
8 }
```

如上所示，一个while循环不停的从阻塞队列中获取数据，取到了就调用SourceContext的collect，把一条数据生产到在Flink环境中，给后面的流程使用；

小结

至此，WikipediaEditsSource源码的分析就完成了，在此小结一下：

1. 和irc.wikimedia.org这个网站建立Socket连接；
2. 连接建立后，读写相关的内容都是基于IRC协议的，这是个应用层的协议，有自己的格式、关键字、命令字等约定，本次分析中我们没有花太多时间在这个协议上，有兴趣的读者在这里了解更多：https://en.wikipedia.org/wiki/Internet_Relay_Chat
3. 启动一个子线程读取Socket信息，收到数据后，构造成WikipediaEditEvent对象，放入阻塞队列中；
4. 原先的那个线程在一个while循环中从阻塞队列中取数据，如果取到了数据就调用ctx.collect方法，这样数据就生产到了Flink环境，其他operator就可以使用了；

以上就是拆解WikipediaEditsSource的过程，现在我们对Flink数据源有了更进一步的了解，后续在开发自定义数据源的时候也有了参考实现；

Read more