StatsDReporter

该类实现了 Scheduled 接口，继承了 AbstractReporter 抽象类，然后里面的属性有：

```
//这个两个常量，代表着 host 和 port，可以通过这两个参数去获取对应的值
public static final String ARG_HOST = "host";
public static final String ARG_PORT = "port";

//代表着这个 reporter 的状态是否是 closed 的，当执行 close 方法时，该值为 true，同时可以根据这个状态来判断是否要把 metric 数据往外发送
private boolean closed = false;

//下面会构造一个 datagram socket
private DatagramSocket socket;

//根据上面获取到的 host 和 port 来构造 InetSocketAddress 对象
private InetSocketAddress address;
```

重写 open 方法，在里面就进行对象的初始化：

```
public void open(MetricConfig config) {
    //根据参数读取配置文件的值
    String host = config.getString(ARG_HOST, null);
    int port = config.getInteger(ARG_PORT, -1);

    //判断是否合法，不合法的话抛出异常
    if (host == null || host.length() == 0 || port < 1) {
        throw new IllegalArgumentException("Invalid host/port
configuration. Host: " + host + " Port: " + port);
    }

    //构建 InetSocketAddress 对象
    this.address = new InetSocketAddress(host, port);

    try {
        //构建 DatagramSocket
        this.socket = new DatagramSocket(0);
    } catch (SocketException e) {
        throw new RuntimeException("Could not create datagram
socket. ", e);
    }
    log.info("Configured StatsDReporter with {host:{}, port:{}}",
host, port);
}
```

重写 report 方法：

```
public void report() {
    //  允许出现异常
    //这样做的目的是为了防止长时间持有锁会阻止运算符的创建和关闭
    try {
        for (Map.Entry<Gauge<?>, String> entry : gauges.entrySet())
{
            if (closed) {
                return;
            }
            //发送 Gauge 数据
            reportGauge(entry.getValue(), entry.getKey());
        }

        for (Map.Entry<Counter, String> entry :
counters.entrySet()) {
            if (closed) {
                return;
            }
            //发送 Counter 数据
            reportCounter(entry.getValue(), entry.getKey());
        }

        for (Map.Entry<Histogram, String> entry :
histograms.entrySet()) {
            //发送 Histogram 数据
            reportHistogram(entry.getValue(), entry.getKey());
        }

        for (Map.Entry<Meter, String> entry : meters.entrySet()) {
            //发送 Meter 数据
            reportMeter(entry.getValue(), entry.getKey());
        }
    }
    catch (ConcurrentModificationException | NoSuchElementException
e) {
        // ignore - may happen when metrics are concurrently added
or removed report next time
    }
}
```

四个发送度量标准数据的方法：

- reportCounter()
- reportGauge()
- reportHistogram()
- reportMeter()

其实可以发现这几个方法内部都是调用的 send 方法，有下面几种：

```
private void send(String name, double value) {
    send(numberIsNegative(value), name, String.valueOf(value));
}

private void send(String name, long value) {
    send(value < 0, name, String.valueOf(value));
}

private void send(boolean resetToZero, String name, String value) {
    if (resetToZero) {
        // negative values are interpreted as reductions instead of
absolute values
        // reset value to 0 before applying reduction as a
workaround
        send(name, "0");
    }
    send(name, value);
}

private void send(final String name, final String value) {
    try {
        String formatted = String.format("%s:%s|g", name, value);
        //将要发送的数据转成字节数组
        byte[] data = formatted.getBytes(StandardCharsets.UTF_8);
        //根据地址将数据发送到其 Socket 中
        socket.send(new DatagramPacket(data, data.length,
this.address));
    }
    catch (IOException e) {
        LOG.error("unable to send packet to statsd at '{}:{}'",
address.getHostName(), address.getPort());
    }
}
```

flink-metrics-statsd 代码就这么多，所以应该是比较简单的，你可以看见这个配置也是比较简单的，你仅需要提供一个 host 和一个 port 就行。

如果你要使用这个 reporter，那么你需要做的就是将 Flink 安装目录下 /opt/flink-metrics-statsd-1.9.0.jar 复制到 lib 目录下，然后你需要在 Flink 的配置文件中配置一下要使用该 reporter，并填写对应的 host 和 port：

```
metrics.reporter.stsd.class:
org.apache.flink.metrics.statsd.StatsDReporter
metrics.reporter.stsd.host: localhost
metrics.reporter.stsd.port: 8125
```

https://ci.apache.org/projects/flink/flink-docs-stable/monitoring/metrics.html#statsd-orgapacheflinkmetricsstatsdstatsdreporter