



中国地质大学

课程报告

课程名称： 计算机网络管理

授课教师： 熊慕舟

学生姓名： 李全可

学生学号： 20171000160

学生班号： 193171

授课日期： 2019. 9-2019. 11

目录

一、 实验题目	3
二、 需求分析	3
(1) 问题描述.....	3
(2) 系统环境.....	3
(3) 实现功能.....	3
三、 模块设计	4
(1) 程序文件说明	4
(2) 总体流程图.....	5
(3) Netty 模块.....	6
(4) Spider 模块	8
(5) Kafka 模块	9
(6) Redis 模块	11
四、 测试结果	12
五、 源程序	18

一、实验题目

利用 netty 库实现一个客户端和服务端，要求每秒的发送次数至少为 10000+

- 客户端 (netty client)
 - 从不同的网站进行爬虫（可以预先设定也可以动态爬虫），并将 html 文本信息存储至 Apache kafka 队列，同时保留 html 的 url 信息。
 - 简单起见可以只爬英文网页。
- 服务端 (netty server)
 - 从 Apache kafka 队列中读取文本信息以及 url 信息。
 - 将接受到的信息保存到 Redis (内含数据库，注意字段划分)。

二、需求分析

（1）问题描述

1. 利用 netty 库建立客户端和服务端，每秒的发送次数至少 10000+ 。
2. Netty 客户端从不同网站爬虫（静态爬虫或动态爬虫），将 html 文本信息和 url 存至 kafka 队列。
3. Netty 服务端从 kafka 队列中读取文本信息以及 url 信息，并将读取的信息保存到 redis 中。

（2）系统环境

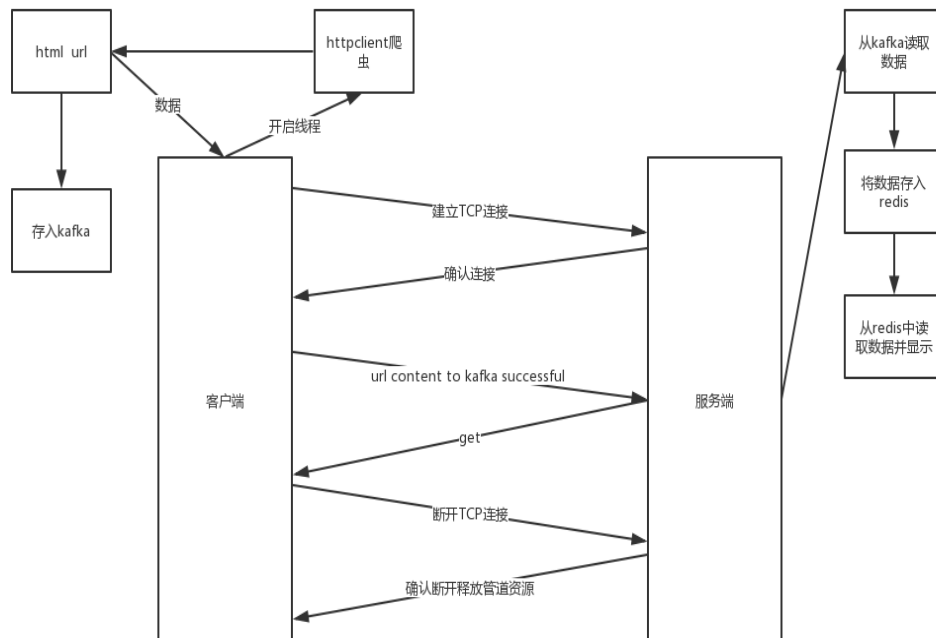
Ubuntu 18.04.3 LTS 操作系统, JavaSE1.8 环境, Eclipse 编译器。

（3）实现功能

1. 使用 netty 库建立服务端和客户端。
2. 客户端开启多线程使用 httpclient 爬取 html 文本信息,将文本信息和对应 url 信息存入 kafka 队列，把爬取过 url 信息发给服务端。
3. 服务端接受到客户端发来 url 信息后，开启多线程从 kafka 队列取出文本信息和对应 url 信息，把文本信息和 url 信息存入 redis 中，并从 redis 中读取数据并显示。
4. 客户端通过多线程（10 个线程）爬虫，爬取 3000 次网站，消耗时间在 60s 左右，速度快，

数据交换次数多。

5. 客户端和服务端之间的交互和操作过程通过输出比较直观和可观。



三、模块设计

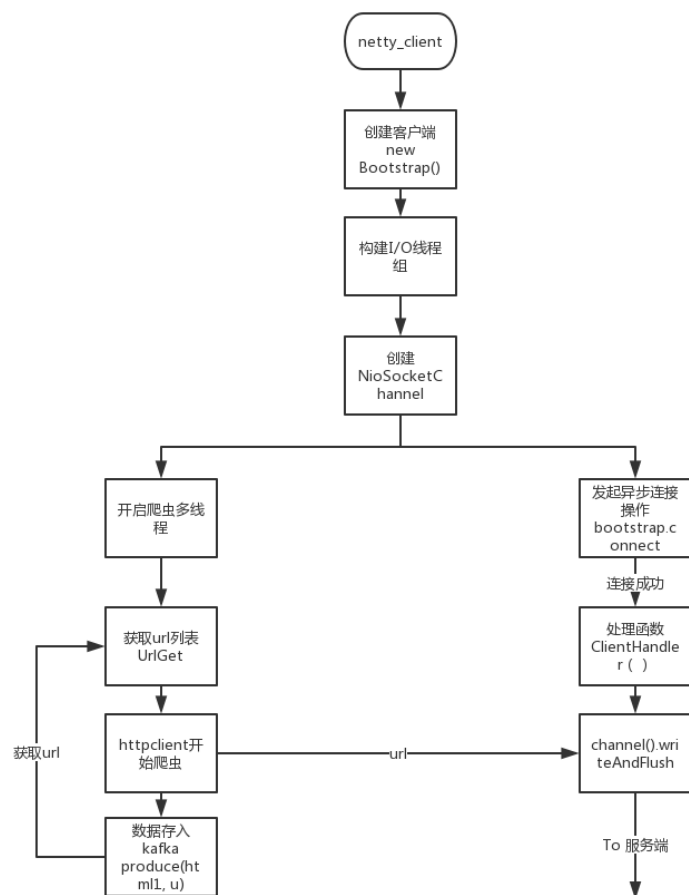
(1) 程序文件说明



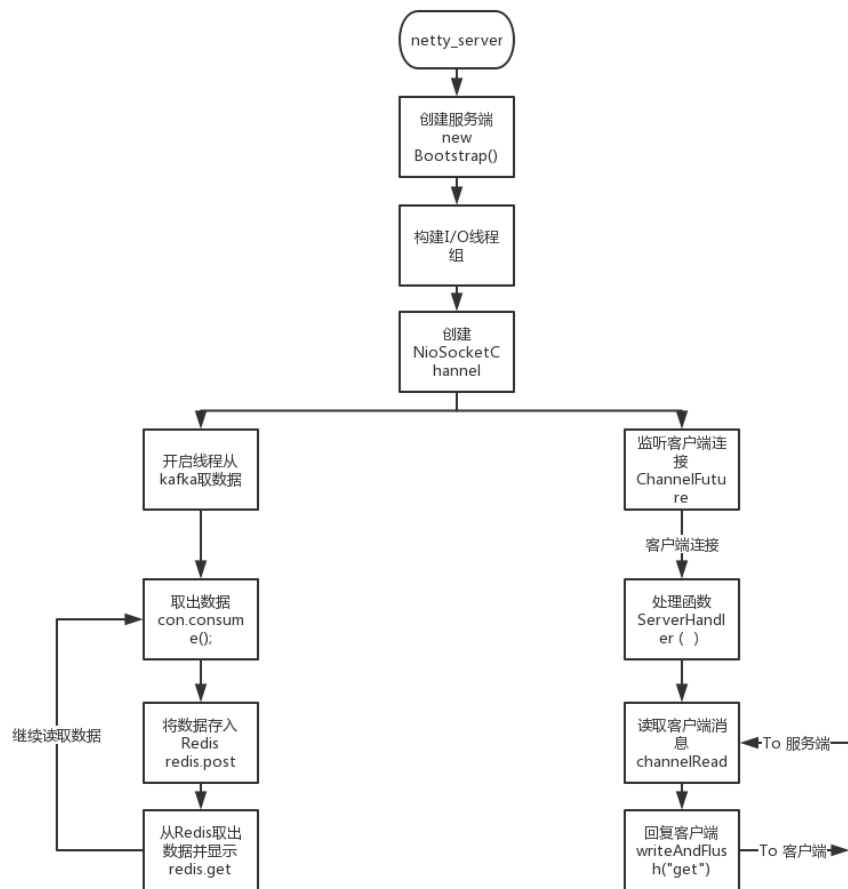
文件名	功能描述
Consumer.java	从 kafka 中取出 html 文本信息和 url 信息
Producer.java	向 kafka 存入 html 文本信息和 url 信息
Netty_Server.java	Netty 服务端，服务端程序入口
Netty_Client.java	Netty 客户端，客户端入口
Redis.java	Redis 操作函数，包含连接 redis 和信息获取、添加、删除。
HttpClient.java	爬取网站 html 文本信息类，包含 http 请求和 html 标签去除
UrlGet.java	获取网站 url 类
log4j.properties	日志输出配置文件
client.txt	Netty 客户端 console 输出结果
server.txt	Netty 服务端 console 输出结果

(2) 总体流程图

客户端总图：



服务端总图：



(3) Netty 模块

介绍：

Netty 是一个 NIO 客户端服务器框架，它支持快速、简单地开发协议服务器和客户端等网络应用程序。它大大简化和简化了网络编程，如 TCP 和 UDP 套接字服务器。“快速而简单”并不意味着最终的应用程序将遭遇可维护性或性能问题。Netty 经过精心设计，积累了许多协议(如 FTP、SMTP、HTTP 和各种二进制和基于文本的遗留协议)实现的经验。因此，Netty 成功地找到了一种方法，可以在不妥协的情况下实现开发、性能、稳定性和灵活性。

重要函数说明：

EventLoopGroup boss = new NioEventLoopGroup(); //boss 线程组监听端口

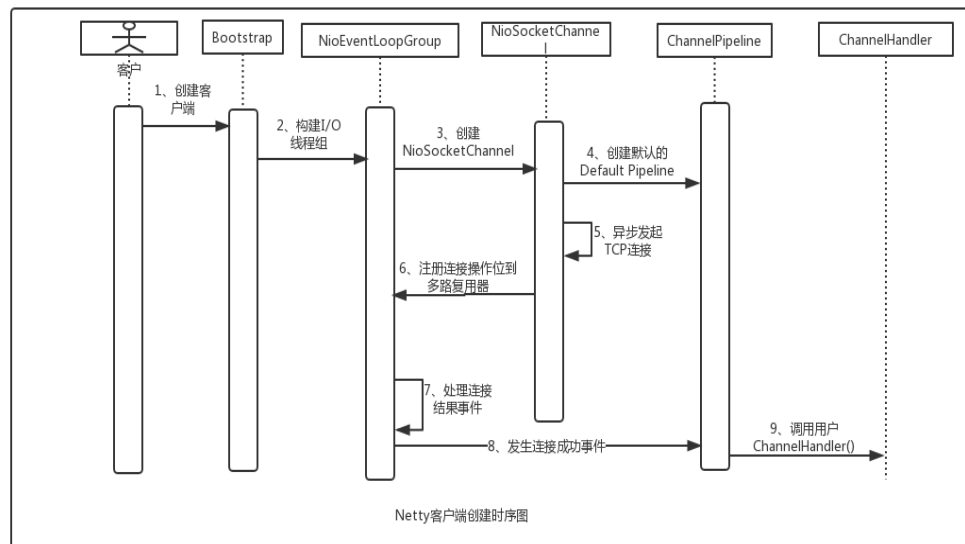
EventLoopGroup worker = new NioEventLoopGroup(); //worker 线程负责数据读写

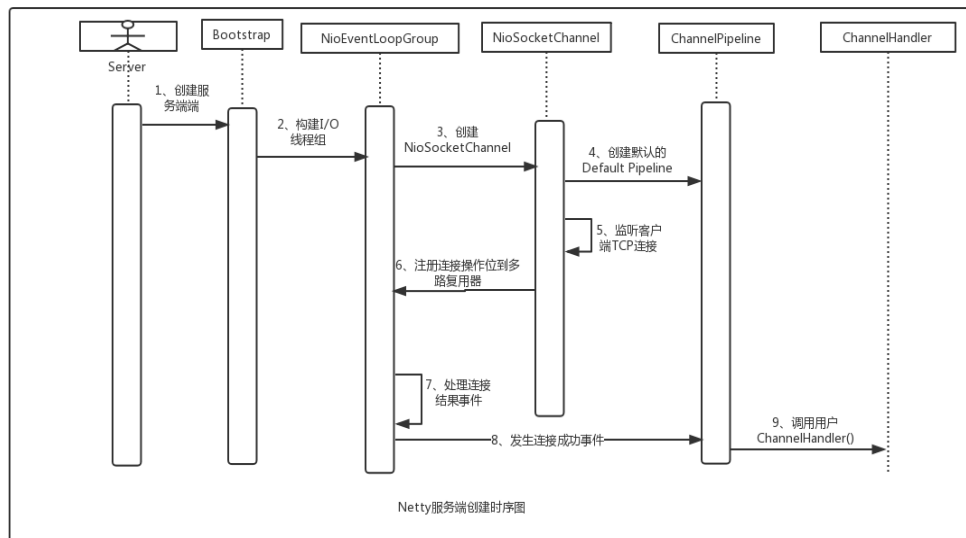
```

ServerBootstrap bootstrap = new ServerBootstrap();    //建立客户端或服务端
bootstrap.group(boss,worker);                        //为客户端或服务端设置线程池
bootstrap.channel(NioServerSocketChannel.class);    //设置 socket 工厂
bootstrap.childHandler(new ChannelInitializer<SocketChannel>()) //设置管道工厂
ChannelPipeline pipeline = socketChannel.pipeline(); //获得管道
pipeline.addLast(new ServerHandler4());            //为管道添加处理类
ChannelFuture future = bootstrap.bind(8866).sync();  //绑定端口
ctx.channel().writeAndFlush("get");                //往管道写入数据
future.channel().closeFuture().sync();              //等待客户端或服务端关闭
boss.shutdownGracefully();                          //关闭线程组
ctx.channel().close();                              //关闭管道
new Thread(new Runnable()                          //创建线程

```

时序图:





(4) Spider 模块

介绍:

HttpClient 是 Apache Jakarta Common 下的子项目，可以用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议，与浏览器在功能上相似。以下列出的是 HttpClient 提供的主要的功能（1）实现了所有 HTTP 的方法（GET,POST,PUT,HEAD 等）（2）支持自动转向（3）支持 HTTPS 协议（4）支持代理服务器等。

重要函数说明:

```

CloseableHttpClient httpClient = HttpClientBuilder.createDefault(); //生成 httpClient

CloseableHttpResponse response = null; //创建 http 响应

HttpGet request = new HttpGet(url); //创建 get 请求

response = httpClient.execute(request); //执行 get 请求

HttpEntity httpEntity = response.getEntity();

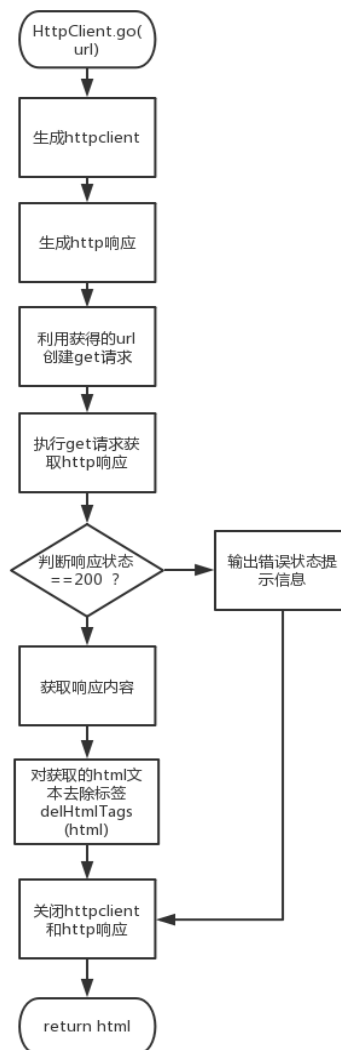
html = EntityUtils.toString(httpEntity, "utf-8"); //获取响应内容

HttpClientUtils.closeQuietly(response); //关闭 http 响应

HttpClientUtils.closeQuietly(httpClient); //关闭 httpClient

htmlStr = htmlStr.replaceAll(scriptRegex, ""); //过滤 html 标签
  
```


流程图：

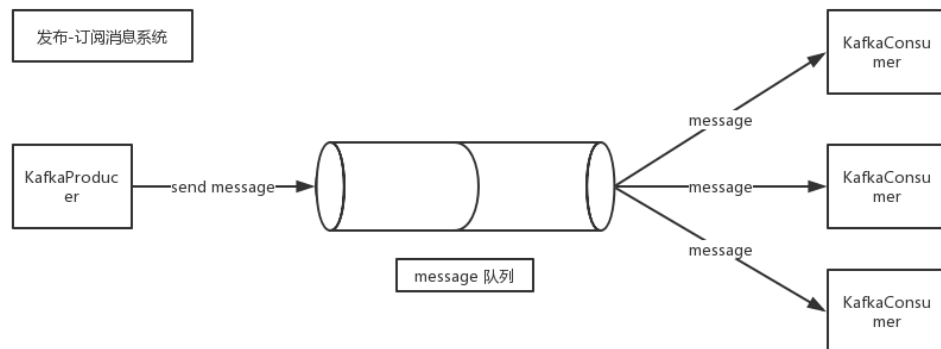
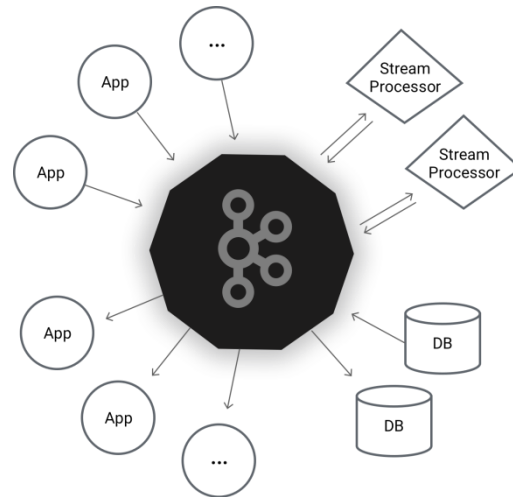


（5）Kafka 模块

介绍：

Apache Kafka 是一个分布式发布 - 订阅消息系统和一个强大的队列，可以处理大量的数据，并使您能够将消息从一个端点传递到另一个端点。Kafka 适合离线和在线消息消费。Kafka 消息保留在磁盘上，并在群集内复制以防止数据丢失。Kafka 构建在 ZooKeeper 同步服务之上。它与 Apache Storm 和 Spark 非常好地集成，用于实时流式数据分析，支持发布 - 订阅消息系统。

Apache Kafka 需要 ZooKeeper 组件，ZooKeeper 用于管理和协调 Kafka 代理。 ZooKeeper 服务主要用于通知生产者和消费者 Kafka 系统中存在任何新代理或 Kafka 系统中代理失败。 根据 Zookeeper 接收到关于代理的存在或失败的通知，然后产品和消费者采取决定并开始与某些其他代理协调他们的任务。



重要函数说明：

```
Properties props = new Properties(); //建立 kafka 连接配置对象
producer = new KafkaProducer<String, String>(props); //建立 kafka 生产者对象
consumer = new KafkaConsumer<String, String>(props); //建立 kafka 消费者对象
producer.send(new ProducerRecord<String, String>(TOPIC,url,data)); //生产者发送
消息至 kafka
ConsumerRecords<String, String> records = consumer.poll(1000); //消费者从 kafka
取出数据
```

```

producer.close(); //关闭生产者
consumer.close(); //关闭消费者者

```

(6) Redis 模块

介绍:

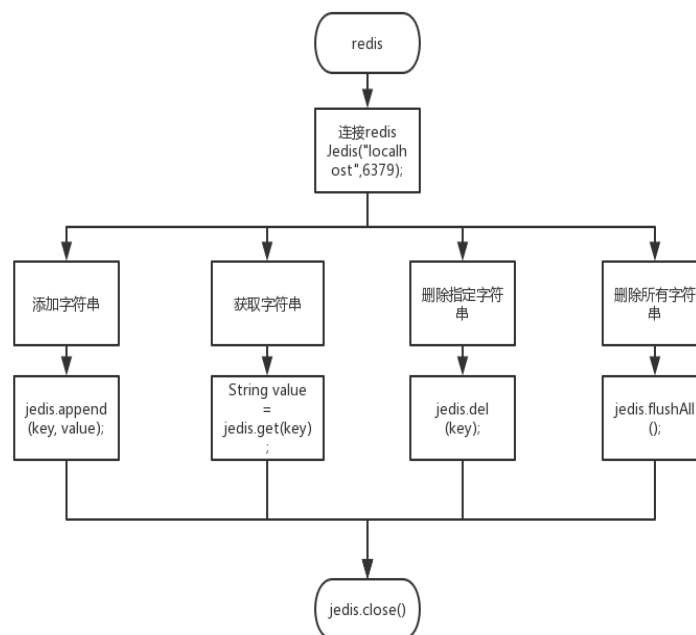
redis 全称: REmote DIctionary Server,是一个支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库、结构化数据的 cache、轻量级的消息队列, key: 消息类型(关键词), value: 消息数据, Redis 支持五种数据类型: string(字符串), hash(哈希), list(列表), set(集合)及 zset(sorted set: 有序集合)。

重要函数说明:

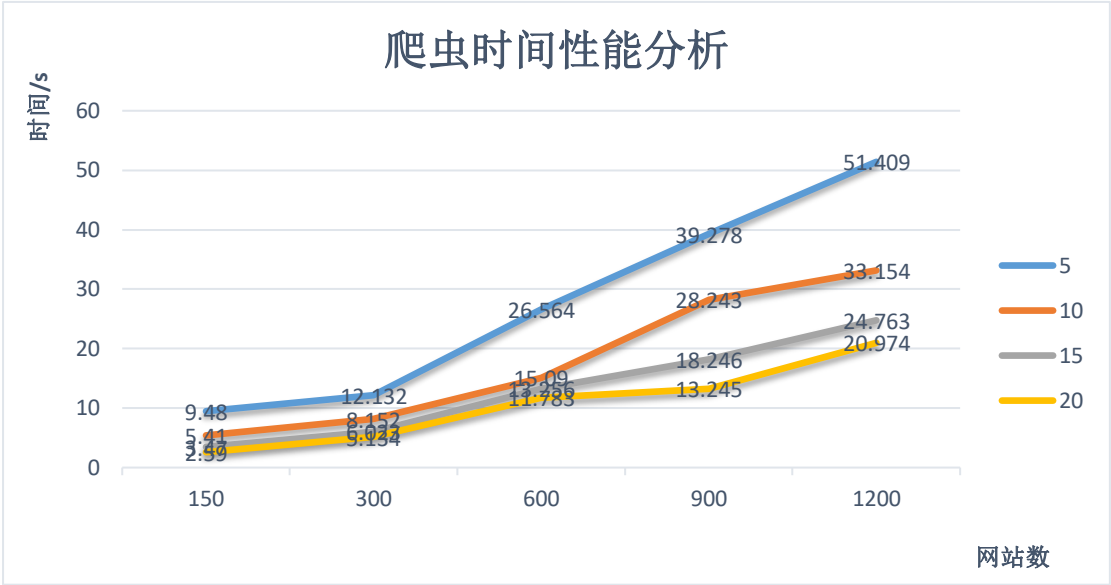
```

Jedis jedis = new Jedis("localhost",6379); //连接 redis 服务并创建对象
jedis.append(key, value); //在指定 key 添加数据 value
String value = jedis.get(key); //获取指定 key 的数据
jedis.del(key); //删除指定 key 的 value
jedis.flushAll(); //删除所有的数据

```



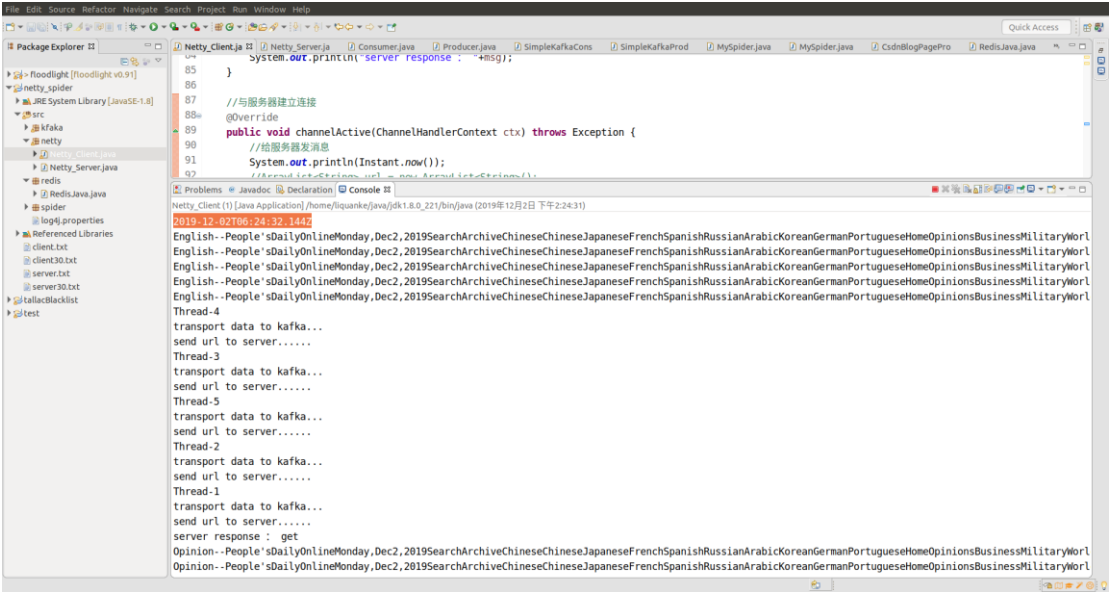
四、测试结果

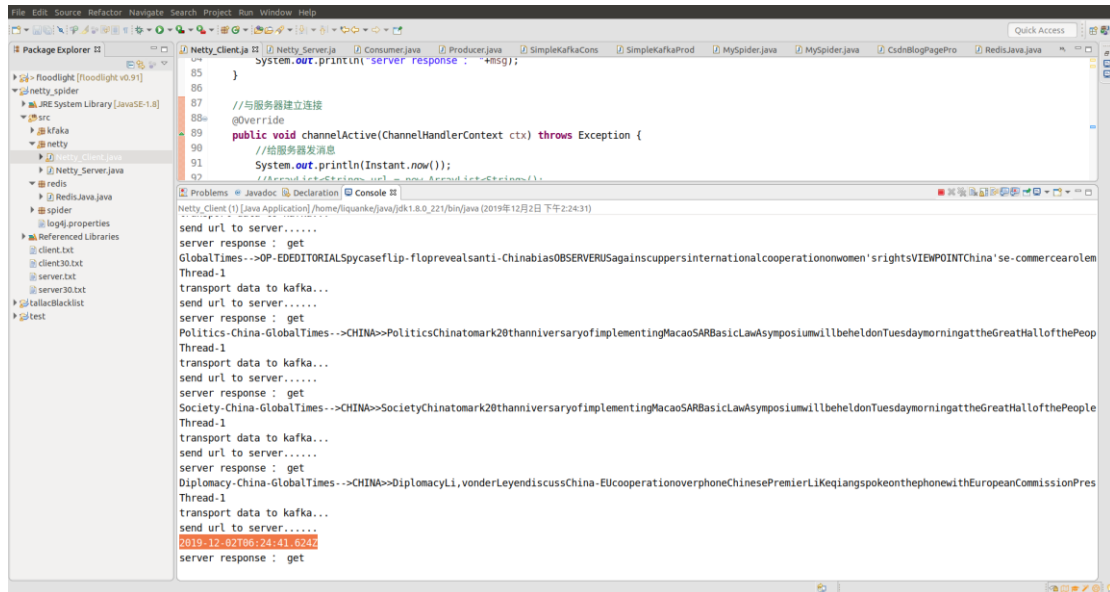


网站数	5 个线程	10 个线程	15 个线程	20 个线程
150	9.48s	5.41s	3.47s	2.59s
300	12.132s	8.152s	6.023s	5.134s
600	26.564s	15.09s	13.256s	11.783s
900	39.278s	28.243s	18.246s	13.245s
1200	51.409s	33.154s	24.763s	20.974s

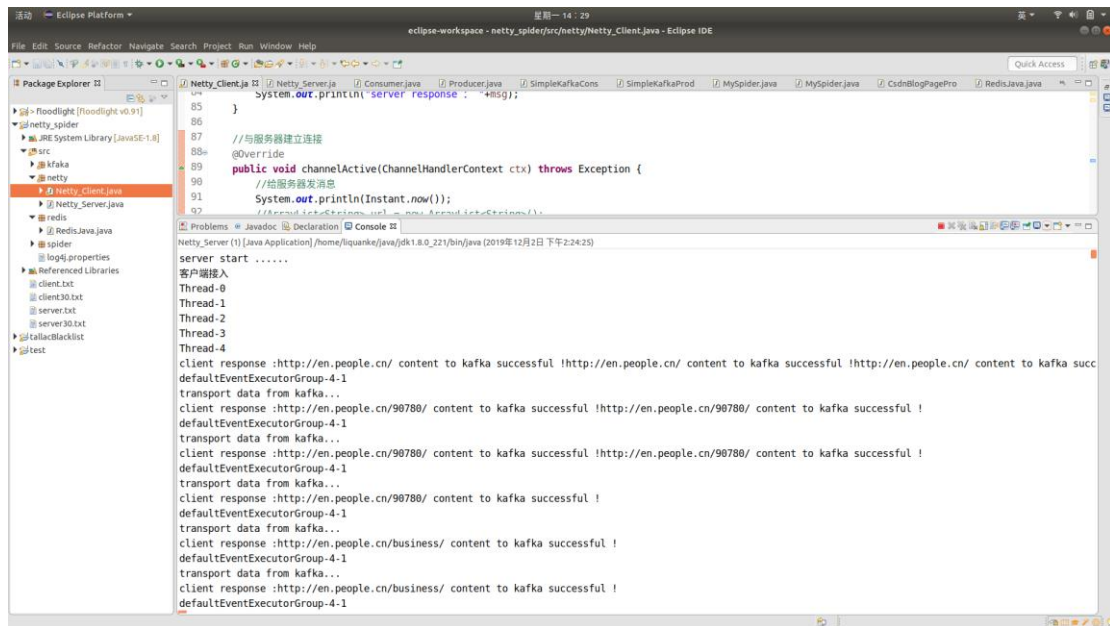
(1)5 个线程爬 150 次网站，开始时间 2019-12-02T06:24:32.144Z，结束时间 2019-12-02T06:24:41.624Z，共耗时 9.48s。

Client:



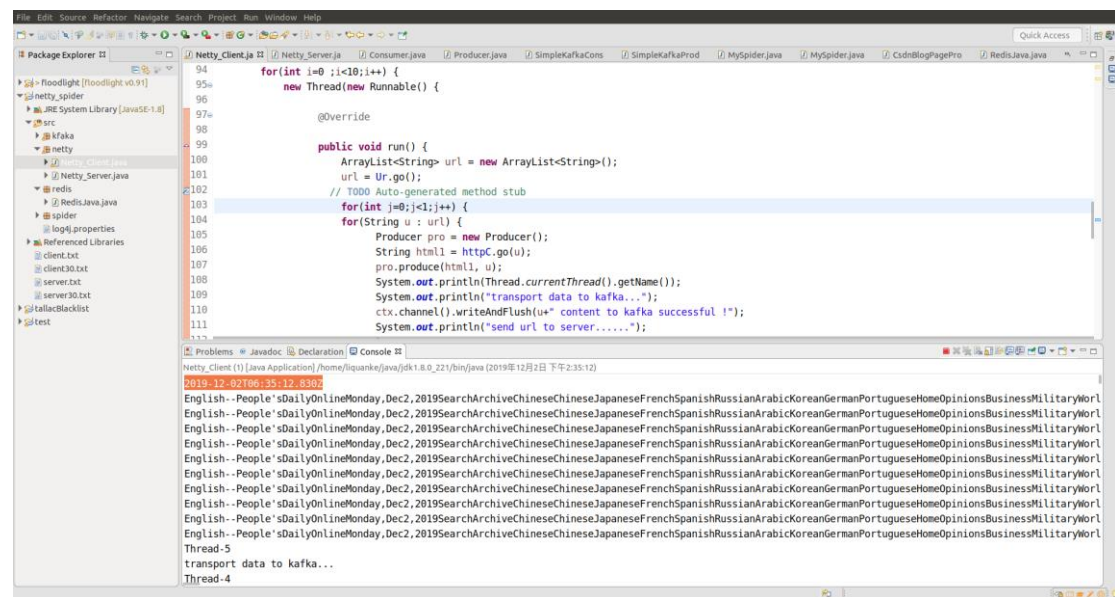


Server:

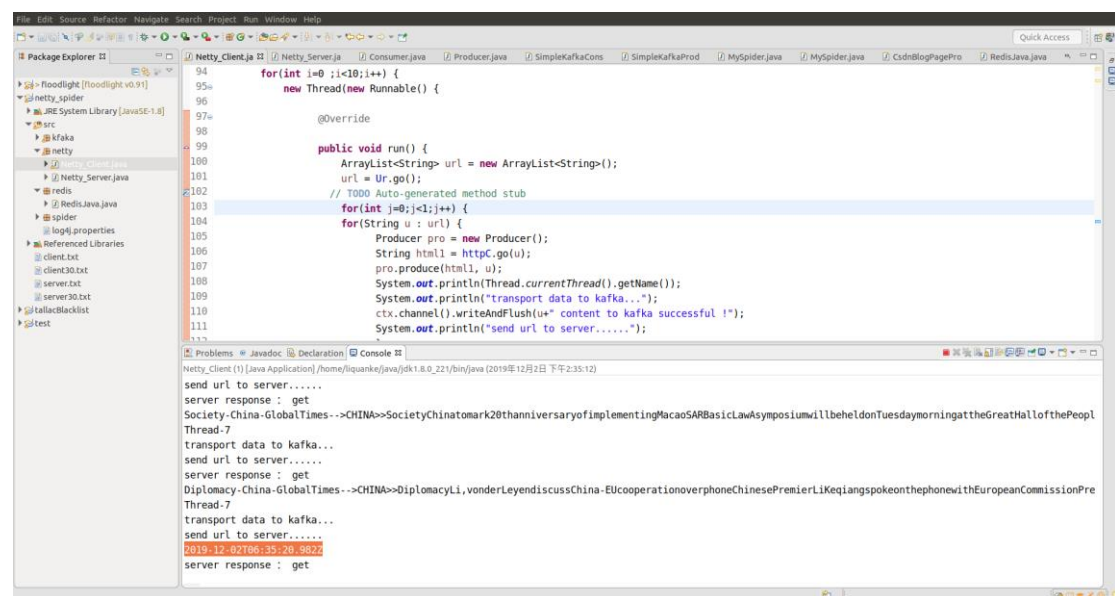


(2)10 个线程爬 300 次网站，开始时间 2019-12-02T06:35:12.830Z，结束时间 2019-12-02T06:35:20.982Z，共耗时 8.152s

Client:

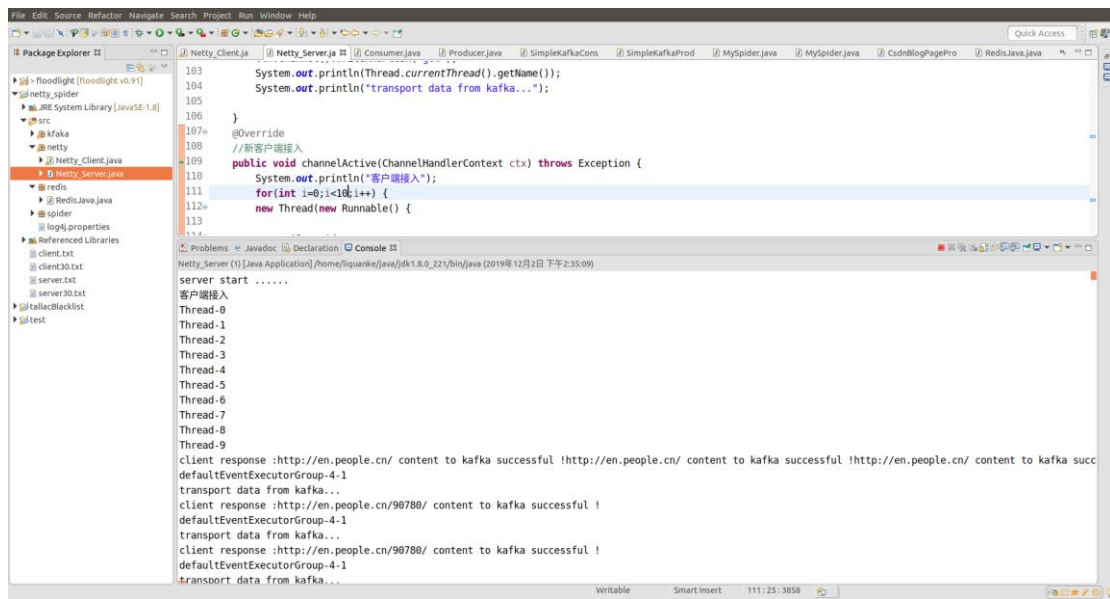


```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  > Floodlight [floodlight v0.91]
  > netty_spider
    > _JRE System Library [javaSE-1.8]
    > src
    > kafka
    > netty
      > Netty_Server.java
      > Redis.java
      > Redis.java.java
      > spider
      > log4j.properties
      > Referenced Libraries
      > client.txt
      > client30.txt
      > server.txt
      > server30.txt
      > talacBlacklist
      > test
Netty_Client.java
  94   for(int i=0 ;i<10;i++) {
  95     new Thread(new Runnable() {
  96
  97       @Override
  98
  99       public void run() {
 100         ArrayList<String> url = new ArrayList<String>();
 101         url = Ur.go();
 102         // TODO Auto-generated method stub
 103         for(int j=0;j<1;j++){
 104           for(String u : url) {
 105             Producer pro = new Producer();
 106             String html1 = httpC.go(u);
 107             pro.produce(html1, u);
 108             System.out.println(Thread.currentThread().getName());
 109             System.out.println("transport data to kafka...");
 110             ctx.channel().writeAndFlush(u+" content to kafka successful !");
 111             System.out.println("send url to server.....");
 112           }
 113         }
 114       }
 115     }
 116   }
Problems Javadoc Declaration Console
Netty_Client (1) [Java Application] /home/liquanke/java/fdk1.8.0_221/bin/java (2019年12月2日 下午2:35:12)
2019-12-02T06:35:12.830Z
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
Thread-5
transport data to kafka...
Thread-4
```

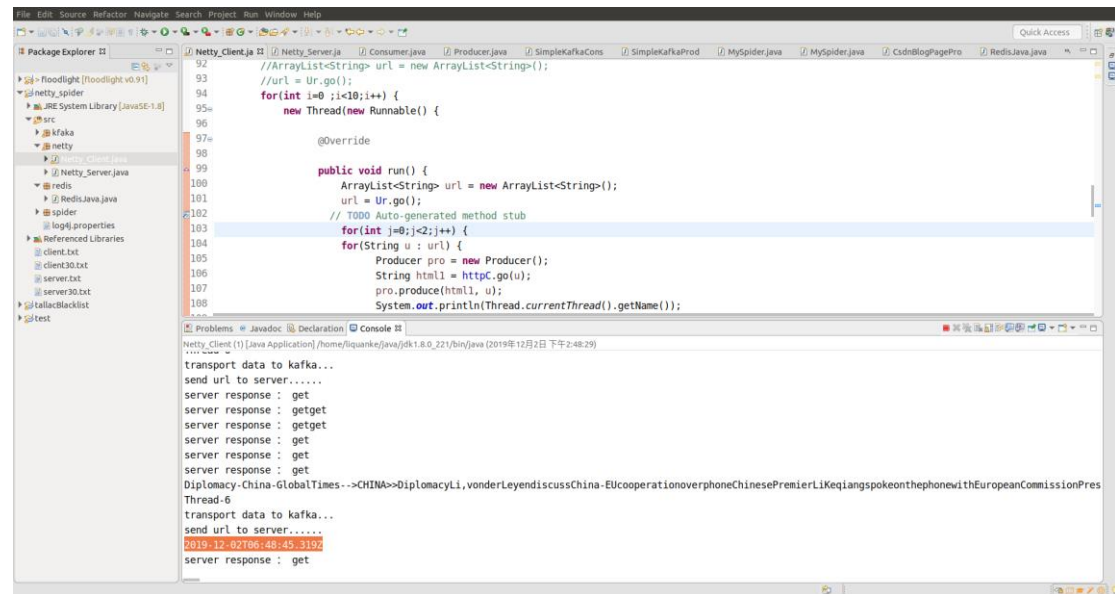


```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  > Floodlight [floodlight v0.91]
  > netty_spider
    > _JRE System Library [javaSE-1.8]
    > src
    > kafka
    > netty
      > Netty_Server.java
      > Redis.java
      > Redis.java.java
      > spider
      > log4j.properties
      > Referenced Libraries
      > client.txt
      > client30.txt
      > server.txt
      > server30.txt
      > talacBlacklist
      > test
Netty_Client.java
  94   for(int i=0 ;i<10;i++) {
  95     new Thread(new Runnable() {
  96
  97       @Override
  98
  99       public void run() {
 100         ArrayList<String> url = new ArrayList<String>();
 101         url = Ur.go();
 102         // TODO Auto-generated method stub
 103         for(int j=0;j<1;j++){
 104           for(String u : url) {
 105             Producer pro = new Producer();
 106             String html1 = httpC.go(u);
 107             pro.produce(html1, u);
 108             System.out.println(Thread.currentThread().getName());
 109             System.out.println("transport data to kafka...");
 110             ctx.channel().writeAndFlush(u+" content to kafka successful !");
 111             System.out.println("send url to server.....");
 112           }
 113         }
 114       }
 115     }
 116   }
Problems Javadoc Declaration Console
Netty_Client (1) [Java Application] /home/liquanke/java/fdk1.8.0_221/bin/java (2019年12月2日 下午2:35:12)
send url to server.....
server response : get
Society-China-GlobalTimes-->CHINA-->SocietyChinaatmark20thanniversaryofImplementingMacaoSARBasicLawAsymposiumwillbeheldonTuesdaymorningattheGreatHallofthePeopl
Thread-7
transport data to kafka...
send url to server.....
server response : get
Diplomacy-China-GlobalTimes-->CHINA-->DiplomacyLi,vonderLeyendiscussChina-EUcooperationoverphoneChinesePremierLiKeqiangspokeonthephonewiththeEuropeanCommissionPre
Thread-7
transport data to kafka...
send url to server.....
2019-12-02T06:35:20.982Z
server response : get
```

Server:



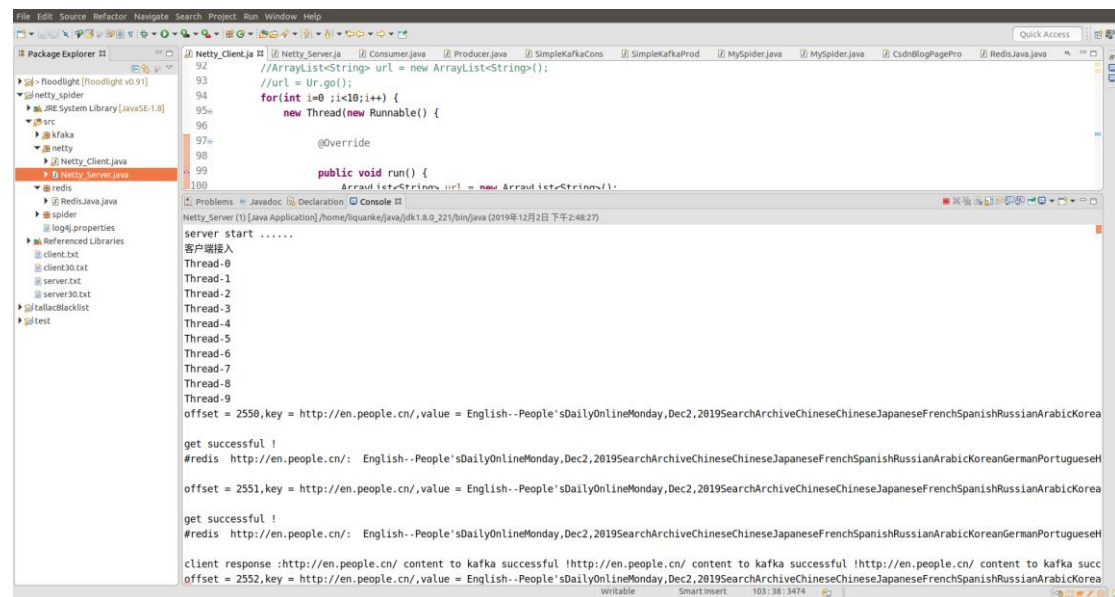
(3) 10 个线程爬 600 次网站, 开始时间 2019-12-02T06:48:30.229Z, 结束时间 2019-12-02T06:48:45.319Z, 共耗时 15.09s



```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  Floodlight [Floodlight v0.91]
  netty_spider
  _JRE System Library [JDK1.8.0_221]
  src
  kafka
  netty
    Netty_Client.java
    Netty_Server.java
  redis
  spider
  log4j.properties
  Referenced Libraries
  client.txt
  client30.txt
  server.txt
  server30.txt
  tailcallblacklist
  test

Netty_Client.java
  92 //ArrayList<String> url = new ArrayList<String>();
  93 //url = Ur.go();
  94 for(int i=0 ;i<10;i++) {
  95     new Thread(new Runnable() {
  96
  97         @Override
  98
  99         public void run() {
  100             ArrayList<String> url = new ArrayList<String>();
  101             url = Ur.go();
  102             // TODO: Auto-generated method stub
  103             for(int j=0;j<2;j++) {
  104                 for(String u : url) {
  105                     Producer pro = new Producer();
  106                     String html1 = httpc.go(u);
  107                     pro.produce(html1, u);
  108                     System.out.println(Thread.currentThread().getName());
  109                 }
  110             }
  111         }
  112     });
  113 }

Problems Javadoc Declaration Console
Netty_Client (1) [Java Application] /home/liquanke/java/jdk1.8.0_221/bin/java (2019年12月2日 下午2:48:29)
transport data to kafka...
send url to server.....
server response : get
server response : getget
server response : getget
server response : get
server response : get
server response : get
Diplomacy-China-GlobalTimes-->CHINA-->DiplomacyLi_vonderLeyendiscussChina-EUcooperationoverphoneChinesePremierLiKeqiangspokeonthephonewithEuropeanCommissionPres
Thread-6
transport data to kafka...
send url to server.....
2019-12-02T06:48:45.319Z
server response : get
```

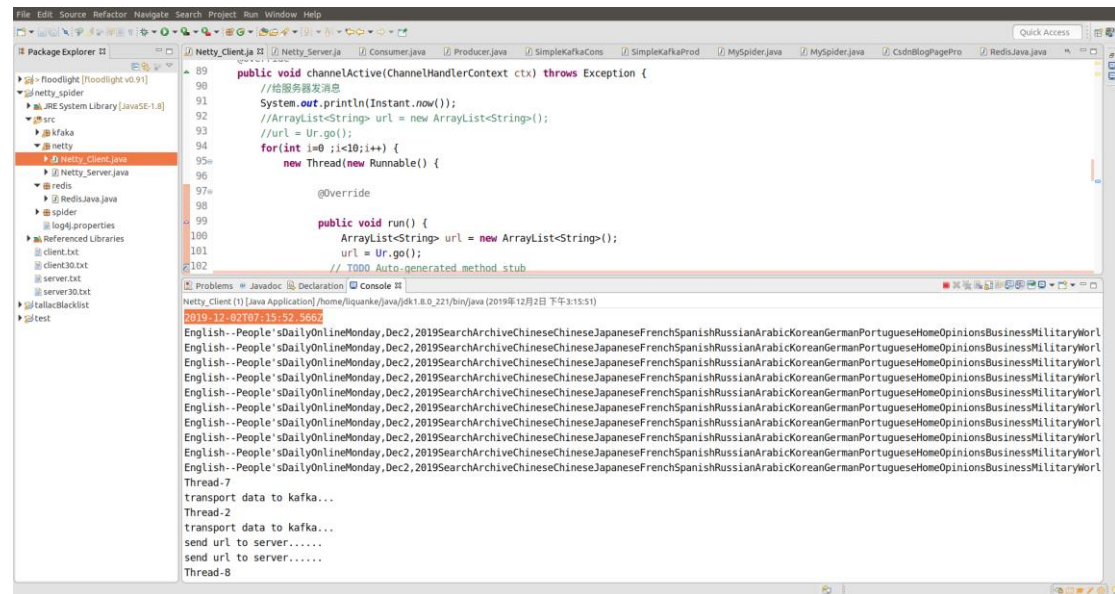


```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  Floodlight [Floodlight v0.91]
  netty_spider
  _JRE System Library [JDK1.8.0_221]
  src
  kafka
  netty
    Netty_Client.java
    Netty_Server.java
  redis
  spider
  log4j.properties
  Referenced Libraries
  client.txt
  client30.txt
  server.txt
  server30.txt
  tailcallblacklist
  test

Netty_Server.java
  92 //ArrayList<String> url = new ArrayList<String>();
  93 //url = Ur.go();
  94 for(int i=0 ;i<10;i++) {
  95     new Thread(new Runnable() {
  96
  97         @Override
  98
  99         public void run() {
  100             ArrayList<String> url = new ArrayList<String>();
  101             url = Ur.go();
  102             // TODO: Auto-generated method stub
  103             for(int j=0;j<2;j++) {
  104                 for(String u : url) {
  105                     Producer pro = new Producer();
  106                     String html1 = httpc.go(u);
  107                     pro.produce(html1, u);
  108                     System.out.println(Thread.currentThread().getName());
  109                 }
  110             }
  111         }
  112     });
  113 }

Problems Javadoc Declaration Console
Netty_Server (1) [Java Application] /home/liquanke/java/jdk1.8.0_221/bin/java (2019年12月2日 下午2:48:27)
server start .....
客户端接入
Thread-0
Thread-1
Thread-2
Thread-3
Thread-4
Thread-5
Thread-6
Thread-7
Thread-8
Thread-9
offset = 2550, key = http://en.people.cn/, value = English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKorea
get successful !
#redis http://en.people.cn/: English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseH
offset = 2551, key = http://en.people.cn/, value = English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKorea
get successful !
#redis http://en.people.cn/: English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseH
client response :http://en.people.cn/ content to kafka successful http://en.people.cn/ content to kafka successful http://en.people.cn/ content to kafka succ
offset = 2552, key = http://en.people.cn/, value = English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKorea
```

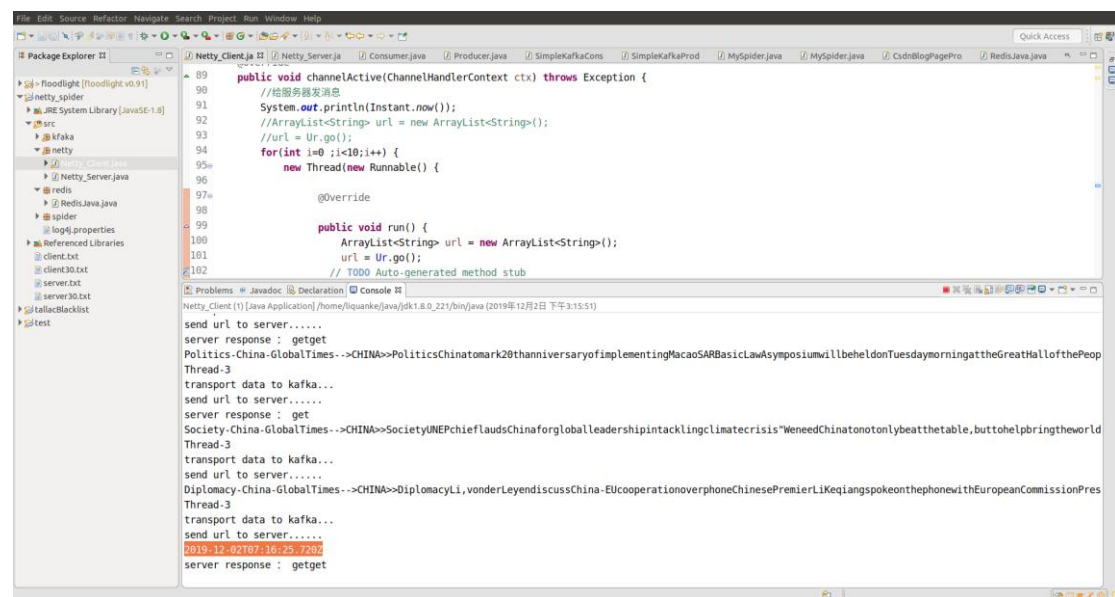

(4)10 个线程爬 1200 次网站，开始时间 2019-12-02T07:15:52.566Z，结束时间 2019-12-02T07:16:25.720Z，共耗时 33.154s



```
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
  > Floodlight [Floodlight v0.9.1]
  > netty_spider
  > _JRE System Library [JavaSE-1.8]
  > src
  > kafka
  > netty
    > Netty_Client.java
    > Netty_Server.java
  > redis
  > Redis.java.java
  > spider
  > log4j.properties
  > Referenced Libraries
  > client.txt
  > client30.txt
  > server.txt
  > server30.txt
  > talacblacklist
  > test
  > test

89 public void channelActive(ChannelHandlerContext ctx) throws Exception {
90     //给服务器发消息
91     System.out.println(Instant.now());
92     //ArrayList<String> url = new ArrayList<String>();
93     //url = Ur.go();
94     for(int i=0 ;i<10;i++) {
95         new Thread(new Runnable() {
96             @Override
97             public void run() {
98                 ArrayList<String> url = new ArrayList<String>();
99                 url = Ur.go();
100                 // TODO Auto-generated method stub
101             }
102         });
103     }
104 }

Problems Javadoc Declaration Console
Netty_Client (1) [Java Application] /home/liquanke/java/jdk1.8.0_221/bin/java (2019年12月2日 下午3:15:51)
2019-12-02T07:15:52.566Z
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
English--People'sDailyOnlineMonday,Dec2,2019SearchArchiveChineseChineseJapaneseFrenchSpanishRussianArabicKoreanGermanPortugueseHomeOpinionsBusinessMilitaryWorld
Thread-7
transport data to kafka...
Thread-2
transport data to kafka...
send url to server.....
send url to server.....
Thread-8
```



```
89 public void channelActive(ChannelHandlerContext ctx) throws Exception {
90     //给服务器发消息
91     System.out.println(Instant.now());
92     //ArrayList<String> url = new ArrayList<String>();
93     //url = Ur.go();
94     for(int i=0 ;i<10;i++) {
95         new Thread(new Runnable() {
96             @Override
97             public void run() {
98                 ArrayList<String> url = new ArrayList<String>();
99                 url = Ur.go();
100                 // TODO Auto-generated method stub
101             }
102         });
103     }
104 }

Problems Javadoc Declaration Console
Netty_Client (1) [Java Application] /home/liquanke/java/jdk1.8.0_221/bin/java (2019年12月2日 下午3:15:51)
send url to server.....
server response : getget
Politics-China-GlobalTimes-->CHINA>>PoliticsChinatamark20thanniversaryofimplementingMacaoSARBasicLawasymposiumwillbeheldonTuesdaymorningattheGreatHallofthePeop
Thread-3
transport data to kafka...
send url to server.....
server response : get
Society-China-GlobalTimes-->CHINA>>SocietyUNEPchief lauds China for global leadership in tackling climate crisis "Wen said China not only beat the table, but to help bring the world
Thread-3
transport data to kafka...
send url to server.....
Diplomacy-China-GlobalTimes-->CHINA>>DiplomacyLi, von der Leyen discuss China-EU cooperation over phone Chinese Premier Li Keqiang spoke on the phone with European Commission Pres
Thread-3
transport data to kafka...
send url to server.....
2019-12-02T07:16:25.720Z
server response : getget
```

五、源程序

netty 包

Netty_Client.java

```
package netty;

import spider.*;

import java.net.InetSocketAddress;
import java.time.Instant;
import java.util.ArrayList;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import io.netty.bootstrap.Bootstrap;
import io.netty.channel.Channel;
import io.netty.channel.ChannelFuture;
import io.netty.channel.ChannelHandler;
import io
io.netty.channel.ChannelHandlerContext;
import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelPipeline;
import io.netty.channel.EventLoopGroup;
import
io.netty.channel.SimpleChannelInboundHandl
er;
import
io.netty.channel.nio.NioEventLoopGroup;
import
io.netty.channel.socket.SocketChannel;
import
io.netty.channel.socket.nio.NioSocketChannel
;
import
io.netty.handler.codec.string.StringDecoder;
import
io.netty.handler.codec.string.StringEncoder;
import
io.netty.util.concurrent.DefaultEventExecutor
Group;
import
io.netty.util.concurrent.EventExecutorGroup;
```

```
import kfaka.Producer;

public class Netty_Client {
    public static void main(String[] args) {
        //worker 负责读写数据
        EventLoopGroup worker = new
NioEventLoopGroup();
        ExecutorService executor =
Executors.newCachedThreadPool();
        executor =
Executors.newFixedThreadPool(9);
        EventExecutorGroup businessGroup
= new DefaultEventExecutorGroup(10);
        try {
            //辅助启动类
            Bootstrap bootstrap = new
Bootstrap();

            //设置线程池
            bootstrap.group(worker);

            //设置 socket 工厂
            bootstrap.channel((Class<?
extends Channel>) NioSocketChannel.class);

            //设置管道
            bootstrap.handler(new
ChannelInitializer<SocketChannel>() {
                @Override
                protected void
initChannel(SocketChannel socketChannel)
throws Exception {
                    //获取管道
                    ChannelPipeline
pipeline = socketChannel.pipeline();
                    //字符串解码器
                    pipeline.addLast(new
StringDecoder());
                    //字符串编码器
                    pipeline.addLast(new
StringEncoder());
                }
            });
        }
    }
}
```

```

//处理类

//pipeline.addLast(new ClientHandler4());

pipeline.addLast(businessGroup,new
ClientHandler4());
    }
});

//发起异步连接操作
ChannelFuture futrue =
bootstrap.connect(new
InetSocketAddress("127.0.0.1",8866)).sync();
//等待客户端链路关闭

futrue.channel().closeFuture().sync();
    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        //优雅的退出，释放 NIO 线程
        worker.shutdownGracefully();
    }
}

}

class ClientHandler4 extends
SimpleChannelInboundHandler<String> {
    UrlGet Ur = new UrlGet();
    MySpider spider = new MySpider();
    HttpClient httpC = new HttpClient();

    //接受服务端发来的消息
    @Override
    protected void
channelRead0(ChannelHandlerContext ctx,
String msg) throws Exception {
        System.out.println("server
response : "+msg);
    }

    //与服务器建立连接
    @Override

```

```

public void
channelActive(ChannelHandlerContext ctx)
throws Exception {
    //给服务器发消息
    System.out.println(Instant.now());
    //ArrayList<String> url = new
ArrayList<String>();
    //url = Ur.go();
    for(int i=0 ;i<20;i++) {
        new Thread(new Runnable() {

            @Override

            public void run() {
                ArrayList<String>
url = new ArrayList<String>();
                url = Ur.go();
                // TODO
                Auto-generated method stub
                for(int j=0;j<2;j++)
                {
                    for(String u : url) {
                        Producer
pro = new Producer();
                        String
html1 = httpC.go(u);

                        pro.produce(html1, u);

                        System.out.println(Thread.currentThread
().getName());

                        System.out.println("transport data to
kafka...");

                        ctx.channel().writeAndFlush(u+" content
to kafka successful !");

                        System.out.println("send url to
server.....");
                    }
                }

                System.out.println(Instant.now());

```

```

        }

    }

    ).start();
}
/*
for(String u : url) {

    for(int i=0 ;i<10;i++) {
        Producer pro = new
Producer();

        String html1 = httpC.go(u);
        pro.produce(html1, u);

        System.out.println(Thread.currentThread
().getName());

        System.out.println("transport
data to kafka...");

        ctx.channel().writeAndFlush(u+" content
to kafka successful !");

        System.out.println("send url to
server.....");
    }

}
*/

}

//与服务端断开连接
@Override
public void
channellInactive(ChannelHandlerContext ctx)
throws Exception {
    System.out.println("断开连接");
}

//异常
@Override
public void
exceptionCaught(ChannelHandlerContext ctx,

```

```

Throwable cause) throws Exception {
    //关闭管道
    ctx.channel().close();
    //打印异常信息
    cause.printStackTrace();
}

public void function(String u) {
    Producer pro = new Producer();
    String html1 = httpC.go(u);
    pro.produce(html1, u);
}
}

Netty_Server.java

package netty;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.ChannelFuture;
import
io.netty.channel.ChannelHandlerContext;
import io.netty.channel.ChannelInitializer;
import io.netty.channel.ChannelOption;
import io.netty.channel.ChannelPipeline;
import io.netty.channel.EventLoopGroup;
import
io.netty.channel.SimpleChannelInboundHandl
er;
import
io.netty.channel.nio.NioEventLoopGroup;
import
io.netty.channel.socket.SocketChannel;
import
io.netty.channel.socket.nio.NioServerSocketC
hannel;
import
io.netty.handler.codec.string.StringDecoder;
import
io.netty.handler.codec.string.StringEncoder;
import
io.netty.util.concurrent.DefaultEventExecutor
Group;
import
io.netty.util.concurrent.EventExecutorGroup;

```

```

import kafka.Consumer;
public class Netty_Server {
    public static void main(String[] args) {

        //boss 线程监听端口，worker 线程
        负责数据读写
        EventLoopGroup boss = new
        NioEventLoopGroup();
        EventLoopGroup worker = new
        NioEventLoopGroup();
        EventExecutorGroup businessGroup
        = new DefaultEventExecutorGroup(10);

        try{
            //辅助启动类
            ServerBootstrap bootstrap =
            new ServerBootstrap();
            //设置线程池
            bootstrap.group(boss,worker);

            //设置 socket 工厂

            bootstrap.channel(NioServerSocketChannel.cl
            ass);

            //设置管道工厂
            bootstrap.childHandler(new
            ChannelInitializer<SocketChannel>() {
                @Override
                protected void
                initChannel(SocketChannel socketChannel)
                throws Exception {
                    //获取管道
                    ChannelPipeline
                    pipeline = socketChannel.pipeline();
                    //字符串解码器
                    pipeline.addLast(new
                    StringDecoder());

                    //字符串编码器
                    pipeline.addLast(new
                    StringEncoder());

                    //处理类

                    //pipeline.addLast(new ServerHandler4());

```

```

        pipeline.addLast(businessGroup,new
        ServerHandler4());
        }
    });

    //设置 TCP 参数
    //1. 链接缓冲池的大小
    (ServerSocketChannel 的设置)

    bootstrap.option(ChannelOption.SO_BACKLO
    G,1024);

    //维持链接的活跃，清除死链
    接(SocketChannel 的设置)

    bootstrap.childOption(ChannelOption.SO_KEE
    PALIVE,true);

    //关闭延迟发送

    bootstrap.childOption(ChannelOption.TCP_N
    ODELAY,true);

    //绑定端口
    ChannelFuture future =
    bootstrap.bind(8866).sync();
    System.out.println("server
    start ..... ");

    /*
    for(int i=0;i<5;i++) {
        new Thread(new Runnable() {

            @Override

            public void run() {

                // TODO
                Auto-generated method stub

                System.out.println(Thread.currentThread().ge
                tName());

                Consumer con =
                new Consumer();

                con.consume();
            }

```

```

        }

        ).start();
    }
    */
    //等待服务端监听端口关闭

future.channel().closeFuture().sync();

    } catch (InterruptedException e) {
        e.printStackTrace();
    } finally {
        //优雅退出，释放线程池资源
        boss.shutdownGracefully();
        worker.shutdownGracefully();
    }
}

}

class ServerHandler4 extends
SimpleChannelInboundHandler<String> {
    @Override
    //读取客户端发送的数据
    protected void
channelRead0(ChannelHandlerContext ctx,
String msg) throws Exception {
        System.out.println("client
response :"+msg);
        ctx.channel().writeAndFlush("get");

        System.out.println(Thread.currentThread().ge
tName());
        System.out.println("transport  data
from kafka...");

    }
    @Override
    //新客户端接入
    public void
channelActive(ChannelHandlerContext ctx)
throws Exception {

```

```

        System.out.println("客户端接入");
        for(int i=0;i<20;i++) {
            new Thread(new Runnable() {

                @Override

                public void run() {

                    // TODO Auto-generated
method stub

                    System.out.println(Thread.currentThread().ge
tName());

                    Consumer con = new
Consumer();

                    con.consume();

                }

            }).start();
        }

        @Override
        //客户端断开
        public void
channelInactive(ChannelHandlerContext ctx)
throws Exception {
            System.out.println("客户端断开");
        }
        @Override
        //异常
        public void
exceptionCaught(ChannelHandlerContext ctx,
Throwable cause) throws Exception {
            //关闭通道
            ctx.channel().close();
            //打印异常
            cause.printStackTrace();
        }
    }

Kfaka 包
Producer.java

```

```

package kfaka;

import
org.apache.kafka.clients.producer.KafkaProducer;
import
org.apache.kafka.clients.producer.ProducerRecord;
//import org.apache.log4j.BasicConfigurator;

import java.util.Properties;

public class Producer {
    private static KafkaProducer<String,
String> producer;
    private final static String TOPIC =
"adienTest2";
    public Producer(){
        Properties props = new Properties();
        props.put("bootstrap.servers",
"localhost:9092");
        props.put("acks", "all");
        props.put("retries", 0);
        props.put("batch.size", 16384);
        props.put("linger.ms", 1);
        props.put("buffer.memory",
33554432);
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        //设置分区类,根据 key 进行数据分区
        producer = new
KafkaProducer<String, String>(props);
    }
    public void produce(String data,String
url){
        producer.send(new
ProducerRecord<String,
String>(TOPIC,url,data));

```

```

        System.out.println(data);
        producer.close();
    }
}

Consumer.java

package kfaka;

import
org.apache.kafka.clients.consumer.ConsumerRecord;
import
org.apache.kafka.clients.consumer.ConsumerRecords;
import
org.apache.kafka.clients.consumer.KafkaConsumer;
//import org.apache.log4j.BasicConfigurator;
//import org.apache.log4j.Logger;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Properties;

import redis.RedisJava;
import spider.UrlGet;

public class Consumer {
    private static KafkaConsumer<String,
String> consumer;
    private final static String TOPIC =
"adienTest2";
    public Consumer(){
        Properties props = new Properties();
        props.put("bootstrap.servers",
"localhost:9092");
        //每个消费者分配独立的组号
        props.put("group.id", "test2");
        //如果 value 合法,则自动提交偏移量
        props.put("enable.auto.commit",
"true");
        //设置多久一次更新被消费消息

```

的偏移量

```
props.put("auto.commit.interval.ms",
"1000");
    //设置会话响应的时间，超过这个
    时间 kafka 可以选择放弃消费或者消费下一
    条消息
    props.put("session.timeout.ms",
"30000");
    //自动重置 offset

props.put("auto.offset.reset","earliest");
    props.put("key.deserializer",

"org.apache.kafka.common.serialization.Strin
gDeserializer");
    props.put("value.deserializer",

"org.apache.kafka.common.serialization.Strin
gDeserializer");

    consumer      =      new
KafkaConsumer<String, String>(props);
    }

    public void consume(){
        RedisJava redis = new RedisJava();
        UrlGet ur = new UrlGet();
        ArrayList<String> url = new
ArrayList<String>();
        url = ur.go();

        consumer.subscribe(Arrays.asList(TOPIC));
        while (true) {
            ConsumerRecords<String,
String> records = consumer.poll(1000);
            for (ConsumerRecord<String,
String> record : records){
                System.out.printf("offset
= %d,key = %s,value
= %s",record.offset(),record.key(),
record.value());

                redis.post(record.key(),
record.value());

                System.out.println();
            }
        }
    }
}
```

```
String      res      =
redis.get(record.key());
        System.out.println("#redis
"+record.key()+": " + res);
        System.out.println();
    }
}

}

public static void main(String[] args) {
    //BasicConfigurator.configure(); //
自动快速地使用缺省 Log4j 环境。
    new Consumer().consume();
}
}
```

redis 包

RedisJava.java

```
package redis;

import redis.clients.jedis.Jedis;

public class RedisJava {
    public static void main(String[] args) {
        Jedis      jedis      =      new
Jedis("localhost",6379);
        // 如果 redis 有 密码 的
        jedis.auth(redis 的密码);需要此语句链接
        System.out.println("connect
successful ! "+jedis.ping());

        jedis.append("first","hello");// 找 到
        此 key 对应的值，在 value 字符串后追加字
        符串 如果此 key 不存在将创建
        String value1 = jedis.get("first");//获
        取指定 key 下的 values 的值
        System.out.println(value1);
        String value2 = jedis.get("second");
        System.out.println(value2);

        //jedis.del("first");//删除指定的 key

        jedis.flushAll();//删除所有的 key
    }
}
```



```

        public void post (String key,String value) {
            if(value != null) {
                Jedis jedis = new
Jedis("localhost",6379);
                jedis.append(key, value);
                System.out.println("post
successful !");
            }
        }
        public String get (String key) {
            Jedis jedis = new
Jedis("localhost",6379);
            String value = jedis.get(key);
            System.out.println("get
successful !");
            return value ;
        }
        public void del (String key) {
            Jedis jedis = new
Jedis("localhost",6379);
            jedis.del(key);
            System.out.println("del
successful !");
        }
        public void delall () {
            Jedis jedis = new
Jedis("localhost",6379);
            jedis.flushAll();
            System.out.println("delall
successful !");
        }
    }

```

spider 包 HttpClient.java

```

package spider;

import java.io.IOException;

import org.apache.http.HttpEntity;
import org.apache.http.HttpStatus;
import
org.apache.http.client.ClientProtocolExceptio

```

```

n;
import
org.apache.http.client.methods.CloseableHttp
Response;
import
org.apache.http.client.methods.HttpGet;
import
org.apache.http.client.utils.HttpClientUtils;
import
org.apache.http.impl.client.CloseableHttpClie
nt;
import
org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class HttpClient {
    public static String delHtmlTags(String
htmlStr) {
        //定义 script 的正则表达式，去除
js 可以防止注入
        String
scriptRegex="<script[^>]*?>[\\s\\S]*?<\\scri
pt>";
        //定义 style 的正则表达式，去除
style 样式,防止 css 代码过多时只截取到 css
样式代码
        String
styleRegex="<style[^>]*?>[\\s\\S]*?<\\style>
";
        //定义 HTML 标签的正则表达式，
去除标签，只提取文字内容
        String htmlRegex="<[^>]+>";
        //定义空格,回车,换行符,制表符
        String spaceRegex = "\\s*|\\t|\\r|\\n";

        // 过滤 script 标签
        htmlStr
        =
htmlStr.replaceAll(scriptRegex, "");
        // 过滤 style 标签
        htmlStr
        =
htmlStr.replaceAll(styleRegex, "");
        // 过滤 html 标签
        htmlStr
        =
htmlStr.replaceAll(htmlRegex, "");

```

```

        // 过滤空格等
        htmlStr =
htmlStr.replaceAll(spaceRegex, "");
        htmlStr = htmlStr.replaceAll(" ", "");
        return htmlStr.trim(); // 返回文本
字符串
    }

    public String go(String url) {
        String html=null;
        //1.生成 httpClient, 相当于该打开
        一个浏览器
        CloseableHttpClient httpClient =
HttpClientUtils.createDefault();
        CloseableHttpResponse response =
null;
        //2.创建 get 请求, 相当于在浏览
        器地址栏输入 网址
        HttpGet request = new HttpGet(url);
        try {
            //3.执行 get 请求, 相当于在
            输入地址栏后敲回车键
            response =
httpClient.execute(request);

            //4.判断响应状态为 200, 进
            行处理

            if(response.getStatusLine().getStatusCode()
            == HttpStatus.SC_OK) {
                //5.获取响应内容
                HttpEntity httpEntity =
response.getEntity();
                html =
EntityUtils.toString(httpEntity, "utf-8");
                html = delHtmlTags(html);
                return html ;
            } else {
                // 如果返回状态不是
                200, 比如 404 (页面不存在) 等, 根据情
                况做处理, 这里略
                System.out.println(" 返回
                状态不是 200");
            }
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            //6.关闭
            HttpClientUtils.closeQuietly(response);

            HttpClientUtils.closeQuietly(httpClient);
        }
        return html ;
    }
}

```

```

System.out.println(EntityUtils.toString(respon
se.getEntity(), "utf-8"));
    }
    } catch (ClientProtocolException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        //6.关闭
        HttpClientUtils.closeQuietly(response);

        HttpClientUtils.closeQuietly(httpClient);
    }
    return html ;
}
}

```

UrlGet .java

```

package spider;

import java.util.ArrayList;

public class UrlGet {
    public ArrayList<String> go(){
        ArrayList<String> url = new
ArrayList<String>();
        url.add("http://en.people.cn/");

        url.add("http://en.people.cn/90780/");

        url.add("http://en.people.cn/business/");
        ;

        url.add("http://en.people.cn/90786/");

        url.add("http://en.people.cn/90777/");

        url.add("http://en.people.cn/90882/");

        url.add("http://en.people.cn/90782/");

        url.add("http://en.people.cn/205040/");
    }
}

```

url.add("http://en.people.cn/202936/");	url.add("http://chinaplus.cri.cn/");
url.add("http://en.people.cn/90779/");	url.add("http://chinaplus.cri.cn/podcast"
url.add("http://en.people.cn/102775/"););
url.add("http://en.people.cn/90783/");	url.add("http://chinaplus.cri.cn/nihao");
url.add("http://en.people.cn/98389/");	url.add("http://chinaplus.cri.cn/america"
);
url.add("http://en.people.cn/n3/2019/1129/c90000-9636527.html");	url.add("http://chinaplus.cri.cn/south-africa");
url.add("http://en.people.cn/n3/2019/1129/c90000-9636533.html");	url.add("http://chinaplus.cri.cn/specials"
);
url.add("http://en.people.cn/n3/2019/1129/c90000-9636764.html");	url.add("http://www.globaltimes.cn/index.html");
url.add("http://en.people.cn/n3/2019/1129/c90000-9636538.html");	url.add("http://www.globaltimes.cn/china/politics/");
url.add("http://en.people.cn/n3/2019/1129/c90000-9636511.html");	url.add("http://www.globaltimes.cn/china/society/");
url.add("http://en.people.cn/n3/2019/1129/c90000-9636675.html");	url.add("http://www.globaltimes.cn/china/diplomacy/");
url.add("http://en.people.cn/n3/2019/1101/c90000-9628770.html");	return url;
	}
	}