大数据之网络爬虫

liangqingyu

Published
with GitBook

Table of Contents

Introduction 0

网络爬虫可以干什么？ 1

爬虫基础入门 2

- HttpClient 2.1
- Jsoup 2.2
- HtmlCleaner 2.3
- HtmlUnit 2.4
- 正则表达式 2.5

网络基础知识 3

- 认识URL 3.1
- 网页所包含的信息 3.2
- HTTP请求 3.3
- 常见的web知识 3.4

爬虫结构设计 4

- 新闻博客类型爬虫结构 4.1
- 微博论坛类型爬虫结构 4.2
- 电商类型爬虫结构 4.3
- 黄页类型爬虫结构 4.4
- 通用型结构 4.5

常用的工具 5

- Redis 5.1
- ActiceMQ 5.2
- Mysql 5.3
- ElasticSearch/Lucene/Solr 5.4
- HIVE/HBASE/Zookpeer 5.5
- MogoDB 5.6
- Portia 5.7

其他 5.8

爬虫知识难点 6

反爬的应对措施 6.1

多网站的抓取 6.2

模拟登陆 6.3

验证码识别 6.4

网络抓包 6.5

开源爬虫解读 7

Nutch 7.1

Heritrix 7.2

Crawler4J 7.3

WebMagic 7.4

Scrapy 7.5

JSpider 7.6

Spiderman 7.7

网络爬虫的实例应用 8

网络舆情分析 8.1

信息传播分析 8.2

资讯聚合 8.3

电商销售情况监控 8.4

征信信息抓取 8.5

其他研究型应用 8.6

其他 8.7

关于

随着大数据技术的兴起，网络爬虫技术也逐渐被开发者们关注，越来越多的人开始研究网络爬虫技术。我有幸参与了几个以网络爬虫为核心技术的项目，对于网络爬虫技术有一些基本的见解，为了帮助更多想学习网络爬虫技术的人，我特此将自己认为有用的知识点记录在此。

本书将从比较基础的知识开始介绍网络爬虫（主要是垂直型网络爬虫）。

First Chapter

GitBook allows you to organize your book into chapters, each chapter is stored in a separate file like this one.

HttpClient是一个用来处理Http请求的工具。API十分丰富，是Java语言开发网络爬虫最常用的工具。HttpClient不同的版本之间API可能会有较大改变，目前最新版本为4.5.1，下面的例子（maven构建）将使用这个版本的HttpClient介绍其基本的用法。进入官网

1.maven构建

在pom.xml中加入HttpClient的dependency

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.1</version>
</dependency>
```

2.基础参数初始化

```
    private HttpClient httpclient;

    public void init() {
        //httpClientConnectionManager = new BasicHttpClientConnecti
        HttpClientBuilder builder = HttpClientBuilder.create();
        SocketConfig socketConfig = SocketConfig.custom().setSoTime
                .build();
        builder.setDefaultSocketConfig(socketConfig);
        builder.setMaxConnPerRoute(10);

        //builder.setRetryHandler();
        //builder.setConnectionManager(httpClientConnectionManager)


        httpclient = builder.build();
    }
```

2.HttpGet请求

```
public String get(String url) {

        HttpGet httpget = new HttpGet(url);

        /**
         * 设置user-agent
         */
        httpget.addHeader(
                "user-agent",
                "Mozilla/5.0 (Windows NT 6.1; WOW64)
                AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.

        int port=9999;// proxy  port

        HttpHost proxy=new HttpHost("your proxy hostName",port);

        /**
         * 设置请求参数，例如超时设置,代理等
         */
        RequestConfig requestConfig = RequestConfig.custom()
                .setConnectTimeout(5000)
                .setSocketTimeout(5000)
                .setProxy(proxy).build();

        httpget.setConfig(requestConfig);

        try {
            /**
             * 请求的执行，返回HttpResponse
             */
            HttpResponse response = httpclient.execute(httpget);

            HttpEntity entity = response.getEntity();

            /**
             * 获取返回的内容String(官方不建议使用EntityUtils)
             */
            String content = EntityUtils.toString(entity);
            return content;
```

```
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
```

2.HttpPost请求

```
public String post(String url,Map<String,Object> params){
        HttpPost httpPost=new HttpPost(url);
        /**
         * 设置user-agent
         */
        httpPost.addHeader(
                "user-agent",
                "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/53


        List <NameValuePair> nvps = new ArrayList <NameValuePair>()
        nvps.add(new BasicNameValuePair("username", "vip"));
        nvps.add(new BasicNameValuePair("password", "secret"));
        try {
            httpPost.setEntity(new UrlEncodedFormEntity(nvps));
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }

        int port=9999;// proxy  port
        HttpHost proxy=new HttpHost("your proxy hostName",port);

        /**
         * 设置请求参数，例如超时设置,代理等
         */
        RequestConfig requestConfig = RequestConfig.custom()
                .setConnectTimeout(5000).setSocketTimeout(5000).set
        httpPost.setConfig(requestConfig);
```

```
        try {
            /**
             * 请求的执行，返回HttpResponse
             */
            HttpResponse response = httpclient.execute(httpPost);

            HttpEntity entity = response.getEntity();

            /**
             * 获取返回的内容String(官方不建议使用EntityUtils)
             */
            String content = EntityUtils.toString(entity);
            return content;
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return null;
    }
```

3.HttpClient的主要作用

HttpClient主要是通过模拟正常的Http请求，或者网页源码。这里获取到的网页源码和在浏览器中通过右键-查看网页源代码看到的源码通常是一样的。这是抓取数据的第一步，接下来就需要从中解析出我们自己需要的数据，解析数据的话，通常可以选择Jsoup、HTMLCleaner、Gson、FastJson、正则等工具。

Jsoup是一款java语言编写的Html解析器，具有非常简单的api，能够使用类似css的选择语法，从Html文档中抽取出各种标签的内容、属性等值，是处理Html网页的一个神器。官方网站；项目源码（github）

1.maven构建

```
<dependency>
    <groupId>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
    <version>1.8.3</version>
</dependency>
```