第3章 数据采集

学习目标

- 了解 HTTP；
- 了解爬虫的基本原理；
- 掌握 HDFS API 的基本使用；
- 熟悉 HttpClient 爬虫的使用方法。

在大数据时代背景下，未被使用的信息比例高达 99.4%，原因很大程度都是由于高价值的信息无法获取采集。因此，如何从大数据中采集出有用的信息已经是大数据发展的关键因素之一，数据采集可视为大数据产业的基石。

数据是开展本书项目重要的基础，本章将以爬虫的方式讲述如何使用 Java 工具获取网络数据。

3.1 知识概要

在编写数据采集程序前，先对网络数据采集所涉及的知识点做简单介绍，以奠定网络数据采集的基础知识。

3.1.1 数据源分类

确定数据采集的数据源是数据处理成功的关键因素，也是数据分析的基础。数据源作为数据处理的最底层，主要有三大类数据，分别是系统日志、网络数据和数据库数据，关于这三类数据的采集介绍如下。

1. 系统日志采集

许多公司的业务平台每天都会产生大量的日志数据。由这些日志信息，我们可以得到很多有价值的数据。通过对这些日志信息进行采集，并对采集的数据进行数据分析，可以挖掘出公司业务平台日志数据中的潜在价值。

2. 网络数据采集

通过网络爬虫和一些网站平台提供的公共 API(如 Twitter 和新浪微博 API)等方式来获取网站上的数据，从而可以将网页中的非结构化数据和半结构化数从网页中提取出来，通

过清洗和转换操作得到结构化的数据，然后存储为结构统一的本地文件数据。

3. 数据库采集

一些企业会使用传统的关系型数据库 MySQL 和 Oracle 等来存储数据。除此之外，NoSQL 数据库 Redis 和 MongoDB 也常用于数据的采集。大多数的企业后台几乎每时每刻都会产生业务数据，这些业务数据会以数据库一行记录的形式被直接写入到数据库中。通过数据库采集系统直接与企业业务后台服务器结合，将企业业务后台每时每刻都在产生大量的业务记录写入到数据库中，最后由特定的处理分析系统进行系统分析。

本章将网络数据作为数据源，通过提取、清洗和转换操作，得到结构化数据，后续将以这些结构化数据为基础进行数据分析。

3.1.2 HTTP 请求过程

在浏览器中输入一个 URL 链接便可以在浏览器页面中浏览该 URL 的页面内容。从输入 URL 链接到浏览页面内容，整个过程是通过浏览器向网站所在服务器发送了一个 HTTP 请求，请求头会包含一些这个请求的信息，服务器接收到请求后进行处理和解析，返回一个 HTTP 响应，浏览器接收返回的响应，响应中包含页面的源代码等内容，浏览器接收到响应后对其内容进行解析，最终将网页内容呈现在浏览器窗口中，如图 3-1 所示。

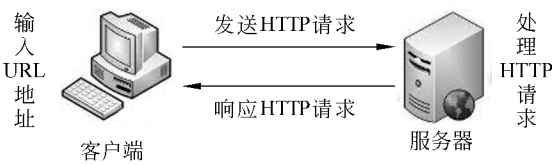

图 3-1 HTTP 请求过程

在 Chrome 浏览器中，通过按 F12 键进入开发者模式，查看 Network 一栏中请求页面的详细内容，如图 3-2 所示。

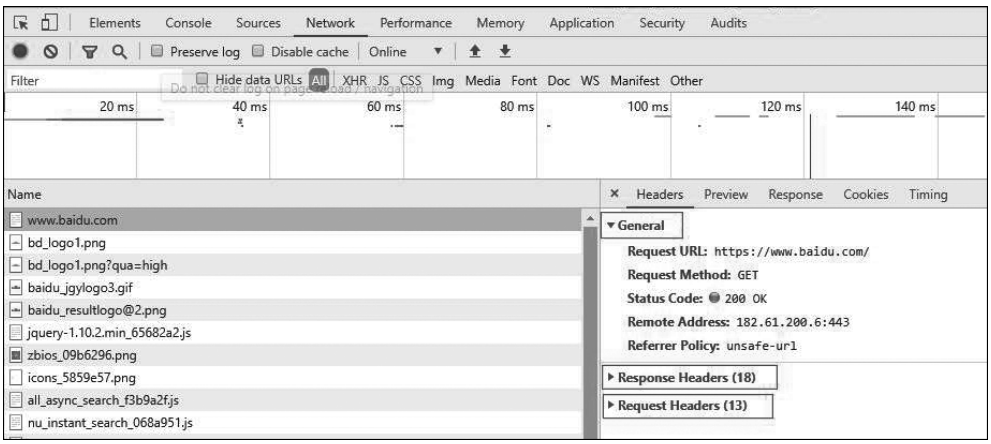

图 3-2 请求页面的详细内容

从图 3-2 中可以看出，General 部分包含 5 个参数，具体介绍如下。

(1) Request URL 参数：请求的 URL 地址。

(2) RequestMethod 参数：请求方法。

(3) Status Code 参数：响应状态码。

(4) Remote Address 参数：远程服务器的地址和端口号。

(5) Referrer Policy 参数：Referrer 判别策略指定该请求是从哪个页面跳转来的，常被用于分析用户来源等信息。

在图 3-2 中有 ResponseHeaders 和 RequestHeaders 两部分内容，它们分别代表响应头和请求头。请求头里带有许多请求信息，例如浏览器标识、Cookies、Host 等信息，服务器会根据请求头内的信息判断请求是否合法，进而做出对应的响应，响应中包含服务器的类型、文档类型、日期等信息，浏览器接收到响应后，会解析响应内容，进而呈现网页内容。接下来，将对 HTTP 请求和响应进行详细介绍。

1. HTTP 请求

HTTP 请求由客户端向服务端发出，可以分为 4 部分内容：请求方法(Request Method)、请求的网址(Request URL)、请求头(Request Headers)、请求体(Request Body)。

(1) 请求方法。

常见的请求方法分为两种：GET 请求和 POST 请求。

在浏览器的地址栏中直接输入 URL 链接可当作发起一次 GET 请求，GET 请求的参数会包含在 URL 链接中。例如，在百度中搜索 Java，链接则变为 https://www.baidu.com/s? wd=Java，其中，URL 中包含请求的参数信息，wd 代表要检索的关键字。POST 请求多用于表单提交，例如，注册用户时，输入用户名、密码和手机号等信息后，单击“注册”按钮，这时客户端通常会向服务端发起一个 POST 请求，这些数据通常以表单的形式传输，而不会出现在 URL 中。

综上所述，在安全性方面，POST 请求的安全性比 GET 请求要高很多，通过 GET 请求提交的数据，用户名和密码都将会以明文的方式出现在 URL 之中，假如注册界面被浏览器缓存，你的账号密码就可以通过查看浏览器历史记录被别人获取到。然而 POST 请求则可以避免出现这种情况。

(2) 请求的网址：指请求地址的 URL 链接。

(3) 请求头。

HTTP 请求头是指在超文本传输协议的请求消息中协议头部分的组件。HTTP 请求头用来准确描述正在获取的资源、服务器或者客户端的行为，定义了 HTTP 事务中的具体操作参数。下面介绍一些常见的 HTTP 请求头。

① Accept：请求报头域，用于指定客户端可接受哪些类型的信息。

② Accept-Language：指定客户端可接受的语言类型。

③ Accept-Encoding：指定客户端可接受的内容编码。

④ Host：用于指定请求资源的主机 IP 和端口号，其内容为请求 URL 的原始服务器或网关的位置。从 HTTP1.1 版本开始，请求必须包含此内容。

⑤ Cookie：也常用复数形式 Cookies，这是网站为了辨别用户进行会话跟踪而存储在用户本地的数据。Cookie 的主要功能是维持当前访问会话。

⑥ Referrer：主要用于标识请求是从哪个页面发过来的，服务器获取到这一信息，做相应的处理，例如，对来源统计和防盗链进行处理操作。

⑦ User-Agent：简称 UA，它是一个特殊的字符串头，可以使服务器识别客户使用的操作系统及版本、浏览器及版本等信息。在做爬取数据时加上此信息，可以伪装为浏览器；如果不加，很可能会被识别为爬取数据。

⑧ Content-Type：也叫互联网媒体类型（Internet Media Type）或者 MIME 类型，在 HTTP 消息头中，它用来表示具体请求中的媒体类型信息。例如，text/html 代表 HTML 格式，image/gif 代表 GIF 图片，application/json 代表 JSON 类型，更多对应关系可以查看此对照表，对照表的网址为 http：//tool.oschina.net/commons。

因此，请求头是请求的重要组成部分，在写爬虫时，大部分情况下都需要设定请求头。

（4）请求体。

请求体通常出现在 POST 请求中，用于存放 POST 请求中的表单数据，而对于 GET 请求而言，请求体为空。

2. HTTP 响应

HTTP 响应由服务器返回给客户端，可以分为三部分：响应状态码（Response Status Code）、响应头（Response Headers）和响应体（Response Body），接下来对这三部分内容进行详细讲解。

1）HTTP 响应状态码

HTTP 响应状态码表示服务器返回给客户端的响应状态，例如，常见的响应代码 200 代表服务器正常响应，404 代表页面未找到，500 代表服务器内部发生错误等，更多 HTTP 响应代码可通过 https://tool.lu/httpcode/进行查看。在爬虫中，可以根据状态码来判断服务器响应状态，如状态码 200，则证明成功返回数据。

2）响应头

响应头包含服务器对客户端请求的应答信息，如 Content-Type、Server、Set-Cookie 等。下面介绍一些常见的 HTTP 响应头。

（1）Date：标识响应产生的时间。

（2）Content-Encoding：指定响应内容的编码。

（3）Server：包含服务器的信息，例如名称、版本号等。

（4）Content-Type：文档类型，指定返回的数据类型是什么，如 text/html 代表返回 HTML 文档，application/x-javascript 代表返回 JavaScript 文件，image/jpeg 则代表返回图片。

（5）Set-Cookie：设置 HTTP Cookie。

（6）Expires：指定响应的过期时间，可以使代理服务器或浏览器将加载的内容更新到缓存中，如果再次访问时，就可以直接从缓存中加载，降低服务器负载，缩短加载时间。

（7）Content-Language：响应体的语言。

3）响应体

最重要的当属响应体的内容了。响应的正文数据都在响应体中，例如请求网页时，它的响应体就是网页的 HTML 代码；请求一张图片时，它的响应体就是图片的二进制数据。我

们做爬虫请求网页后，要解析的内容就是响应体。

3.1.3　认识 HttpClient

HttpClient 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 的客户端编程工具包，并且它支持 HTTP 最新的版本和建议。HttpClient 已经应用在很多的项目中，例如 Apache Jakarta 上很著名的另外两个开源项目 Cactus 和 HTMLUnit 都使用了 HttpClient。

使用 HttpClient 发送请求、接收响应很简单，一般需要如下几步。

(1) 创建 HttpClient 对象。

(2) 创建请求方法的实例，并指定请求 URL。如果需要发送 GET 请求，创建 HttpGet 对象；如果需要发送 POST 请求，创建 HttpPost 对象。

(3) 如果需要发送请求参数，可调用 HttpGet、HttpPost 共同的 setParams(HttpParams params)方法来添加请求参数；对于 HttpPost 对象而言，也可调用 setEntity(HttpEntity entity)方法来设置请求参数。

(4) 调用 HttpClient 对象的 execute(HttpUriRequest request)发送请求，该方法返回一个 HttpResponse。

(5) 调用 HttpResponse 的 getAllHeaders()、getHeaders(String name)等方法可获取服务器的响应头；调用 HttpResponse 的 getEntity()方法获取 HttpEntity 对象，该对象包装服务器的响应内容，程序可通过该对象获取服务器的响应内容。

(6) 释放连接。无论执行方法是否成功，都必须释放连接。

3.2　分析与准备

通过 3.1 节内容了解到网络数据采集的一些基础知识，帮助我们从理论知识方面了解网络数据采集，本节主要对要采集的数据结构进行分析以及创建编写数据采集程序的环境，为最终编写数据采集程序做准备。

3.2.1　分析网页数据结构

在爬取网站数据前要先通过分析网站的源码结构制定爬虫程序的编写方式，以便能获取准确的数据。

使用 Google 浏览器进入开发者模式，切换到 Network 这一项，在浏览器的地址栏中输入要爬取数据网站的 URL，在职位搜索栏中输入想要分析的职位进行检索，这时候可以看到服务器返回的内容，因为内容较多，不太容易找到职位信息数据，所以通过设置过滤，过滤掉不需要的信息，如图 3-3 所示。

因为该网站的职位信息并不在 HTML 源代码里，而是保存在 JSON 文件里，因此在图 3-3 中的过滤一栏中选择 XHR(XML Http Request)过滤规则，这样就可以只看到 Ajax 请求中的 JSON 文件了，我们将要获取的职位信息数据也在这个 JSON 文件中，如图 3-4 所示。

单击图 3-4 中 positionAjax.json 这一条信息，在弹出的窗口选择 Preview 选项，通过逐

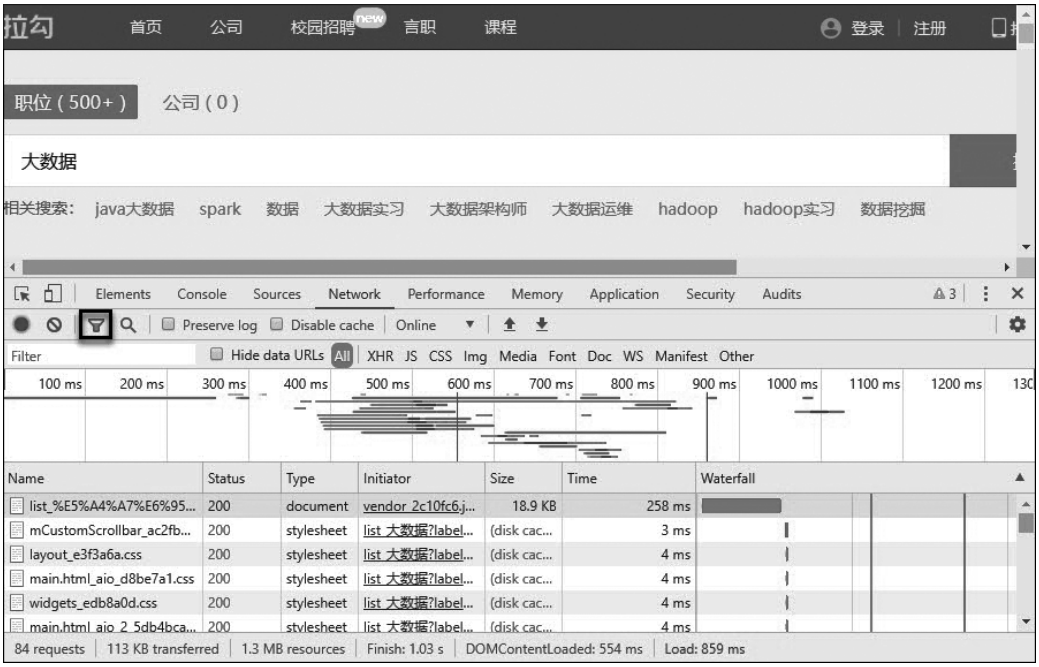

图 3-3 查看网站 Network 信息

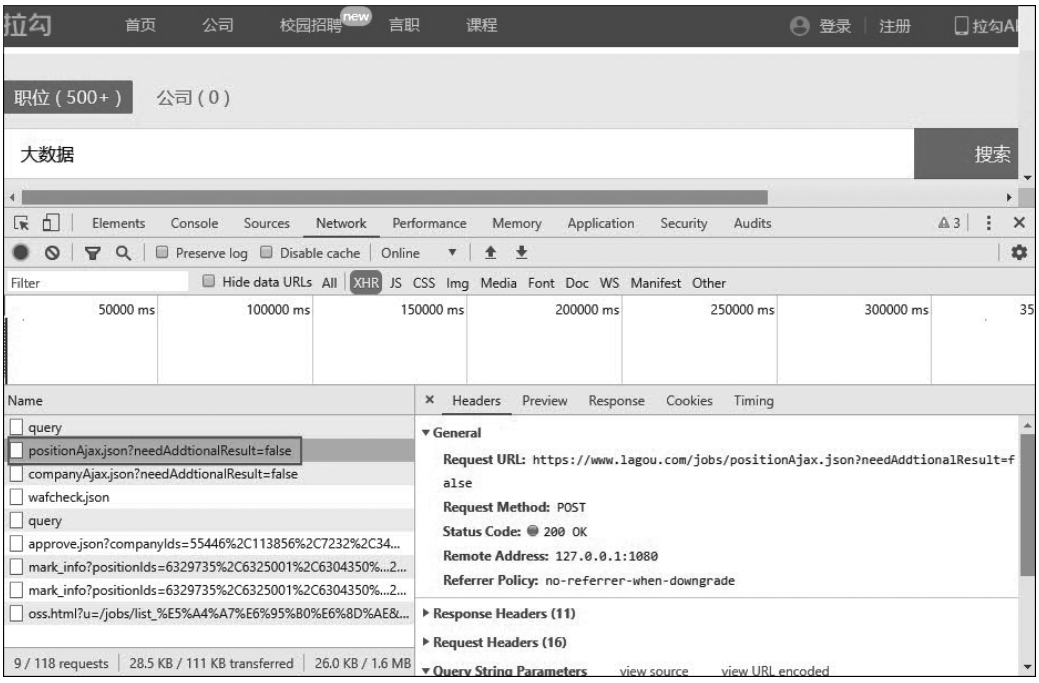

图 3-4 设置过滤内容

级展开 JSON 文件中的数据，在 content→positionResult→result 下查看大数据相关的职位信息，如图 3-5 所示。

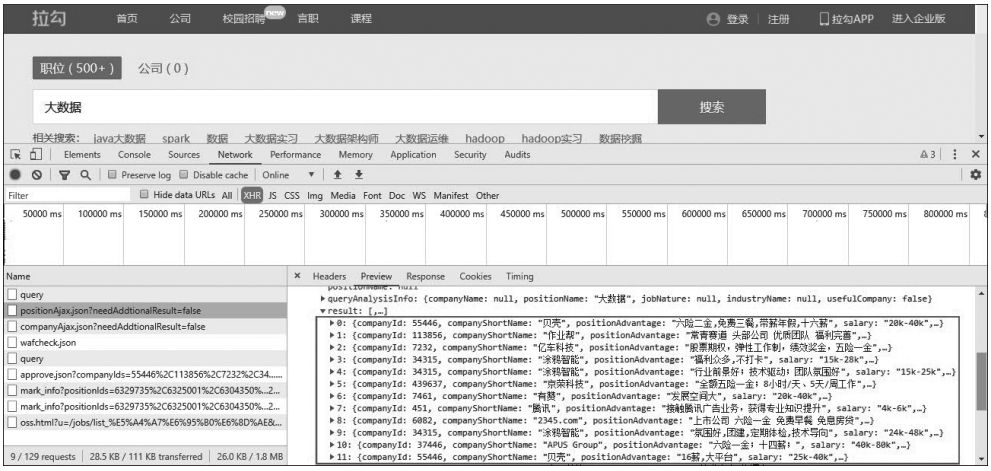

图 3-5　查看大数据相关的职位信息

3.2.2　数据采集环境准备

本章编写数据采集程序主要通过 Eclipse 开发工具完成，本节将详细讲解如何通过 Eclipse 工具编程，实现网络数据的采集。

（1）打开 Eclipse 工具，单击 File→New→Other，进入 Select a wizard 界面，选择要创建工程的类别，这里选择的是 Maven Project，即创建一个 Maven 工程，具体如图 3-6 所示。

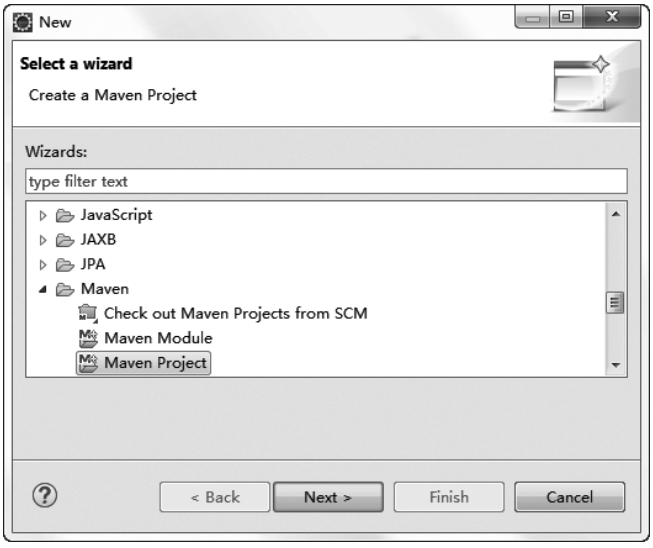

图 3-6　创建 Maven 项目

（2）在图 3-6 中选择创建 Maven Project，单击 Next 按钮，进入新建项目类别的选择界面，勾选 Create a simple project 复选框，创建一个简单的 Maven 工程，具体如图 3-7 所示。

（3）在图 3-7 中，单击 Next 按钮，进入 Maven 工程的配置界面，即指定 Group Id 为 com.itcast.jobcase，指定 Artifact Id 为 jobcase-reptile，并在 Packaging 下拉选项框中选择打包方式为 jar，如图 3-8 所示。

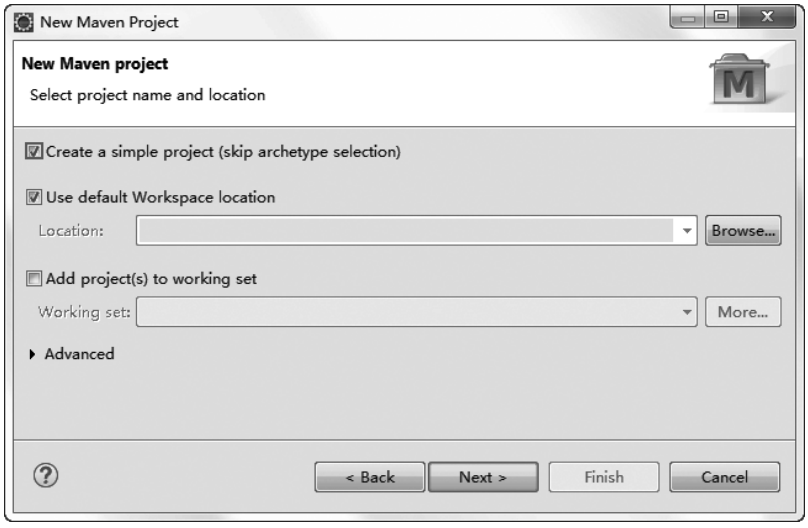

图 3-7 选择创建一个简单的 Maven 工程

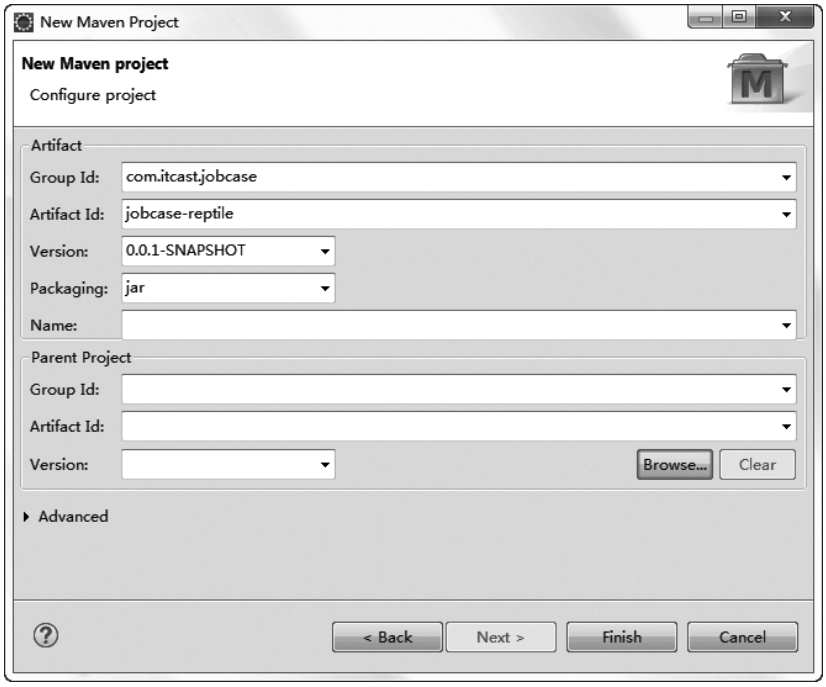

图 3-8 配置 Maven 工程

(4) 在图 3-8 中,单击 Finish 按钮,完成 Maven 工程的配置,创建好的 Maven 工程 jobcase-reptile 如图 3-9 所示。

(5) 在图 3-9 中,双击 jobcase-reptile 工程,选中 src/main/java 文件夹,右键单击 New→Package 创建 Package 包,并命名包名为 com.position.reptile,如图 3-10 所示。

(6) 在图 3-10 中,单击 Finish 按钮,完成 Package 包的创建,创建好的 Package 包如图 3-11 所示。

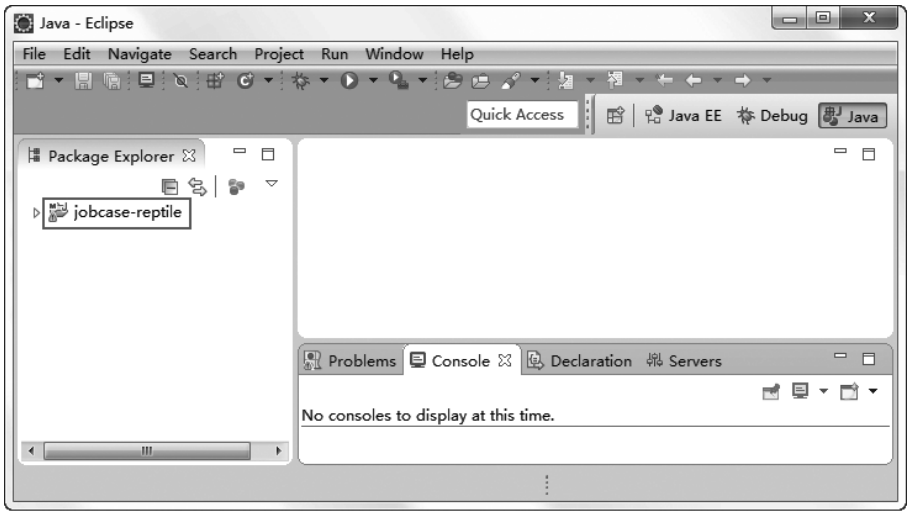

图 3-9　Maven 工程 jobcase-reptile

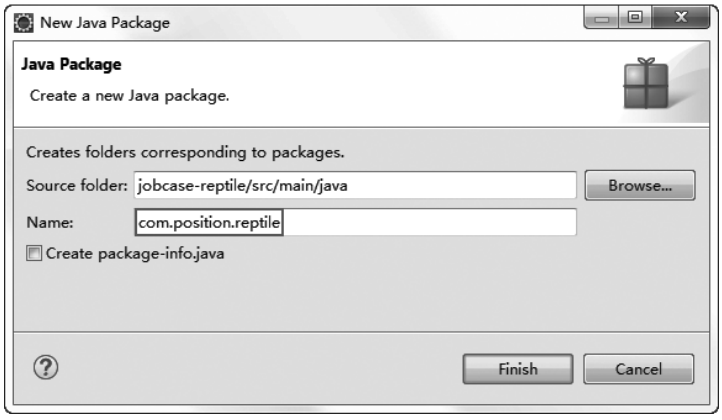

图 3-10　创建 Package 包

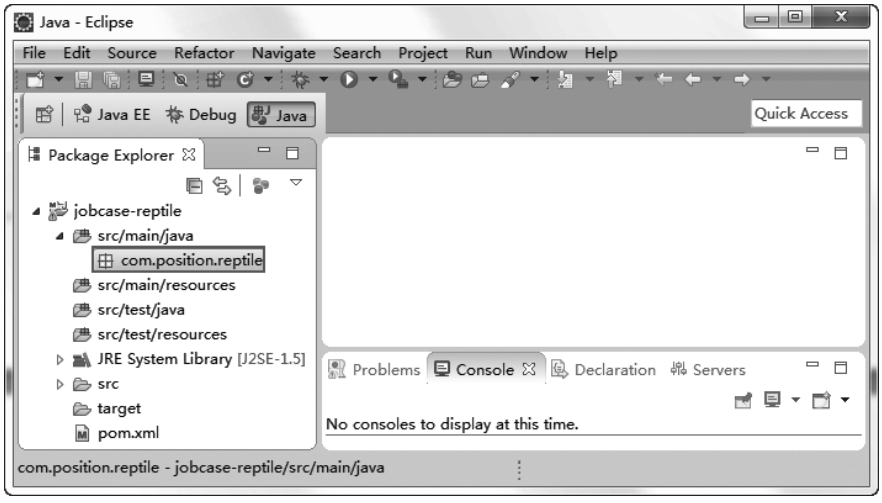

图 3-11　com.position.reptile 包

(7) 在图 3-11 中,双击 pom.xml 文件,添加编写爬虫程序所需的 HttpClient 和 JDK 1.8 依赖。pom.xml 文件添加的内容,具体如文件 3-1 所示。

文件 3-1 pom.xml

```
<project xmlns=http://maven.apache.org/POM/4.0.0
         xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation=
  "http://maven.apache.org/POM/4.0.0  http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
    <groupId>com.itcast.jobcase</groupId>
    <artifactId>jobcase-reptile</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
            <version>4.5.4</version>
        </dependency>
        <dependency>
            <groupId>jdk.tools</groupId>
            <artifactId>jdk.tools</artifactId>
            <version>1.8</version>
            <scope>system</scope>
            <systemPath>${JAVA_HOME}/lib/tools.jar</systemPath>
        </dependency>
    </dependencies>
</project>
```

(8) 选中 jobcase-reptile 工程,右键单击选择 Maven→Update Project 更新工程。至此,就完成了 Maven 工程的搭建。

3.3 采集网页数据

通过前两节的学习,了解了数据采集相关的基础内容,并构建了开展数据采集的基本环境,在后续一节中将详细讲解通过 Java 语言编写基于 HttpClient 的数据采集程序。

3.3.1 创建响应结果 JavaBean 类

本项目采集的网页数据为 HTTP 请求过程中的响应结果数据,通过创建的 HttpClient 响应结果对象作为数据存储的载体,对响应结果中的状态码和数据内容进行封装。

在 com.position.reptile 包下,创建名为 HttpClientResp.java 文件的 JavaBean 类,如文件 3-2 所示。

文件 3-2 HttpClientResp.java

```
import java.io.Serializable;
public class HttpClientResp implements Serializable  {
    private static final long serialVersionUID =2168152194164783950L;
    //响应状态码
```

```
    private int code;
    //响应数据
    private String content;
    //定义无参和有参的构造方法
    public HttpClientResp() {
    }
    public HttpClientResp(int code) {
        this.code =code;
    }
    public HttpClientResp(String content) {
        this.content =content;
    }
    public HttpClientResp(int code, String content) {
        this.code =code;
        this.content =content;
    }
    //定义属性的 get/set 方法
    public int getCode() {
        return code;
    }
    public void setCode(int code) {
        this.code =code;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content =content;
    }
    //重写 toString 方法
    @Override
    public String toString() {
        return "[code=" +code +", content=" +content +"]";
    }
}
```

3.3.2　封装 HTTP 请求的工具类

在 com.position.reptile 包下，创建一个命名为 HttpClientUtils.java 文件的工具类，用于实现 HTTP 请求的方法。

(1) 在类中定义三个全局常量，便于在类中的方法统一访问，下面是这三个常量的概述。

① ENCODING：表示定义发送请求的编码格式。

② CONNECT_TIMEOUT：表示设置建立连接超时时间。

③ SOCKET_TIMEOUT：表示设置请求获取数据的超时时间。

为了有效地防止程序阻塞，可以在程序中设置 CONNECT_TIMEOUT 和 SOCKET_TIMEOUT 这两项参数，通过在类中定义常量的方式定义参数的内容，如文件 3-3 所示。

文件 3-3 HttpClientUtils.java

```
1  //编码格式,发送编码格式统一用 UTF-8
2  private static final String ENCODING ="UTF-8";
3  //设置连接超时时间,单位毫秒
4  private static final int CONNECT_TIMEOUT =6000;
5  //请求获取数据的超时时间(即响应时间),单位毫秒
6  private static final int SOCKET_TIMEOUT =6000;
```

(2) 编写 packageHeader()方法。

在工具类中定义 packageHeader()方法用于封装 HTTP 请求头的参数,如 Cookie、User-Agent 等信息,该方法中包含两个参数,分别为 params 和 httpMethod,如文件 3-4 所示。

文件 3-4 HttpClientUtils.java

```
1 public static void packageHeader(Map<String, String>params,
2 HttpRequestBase httpMethod) {
3     //封装请求头
4     if (params !=null) {
5     /**
6      * 通过 entrySet()方法从 params 中返回所有键值对的集合,并保存在 entrySet 中
7      * 通过 foreach()方法每次取出一个键值对保存在一个 entry 中
8      */
9     Set<Entry<String, String>>entrySet =params.entrySet();
10    for (Entry<String, String>entry : entrySet) {
11    //通过 entry 分别获取键-值,将键-值参数设置到请求头 HttpRequestBase 对象中
12        httpMethod.setHeader(entry.getKey(), entry.getValue());
13        }
14    }
15 }
```

文件 3-4 中的 params 参数的数据类型为 Map<String,String>,主要用于封装请求头中的参数,其中,Map 的 Key 表示请求头参数的名称,Value 代表请求头参数的内容。例如,在请求头中加入参数 Cookie,那么 Map 的 Key 则为 Cookie,Value 则为 Cookie 的具体内容。

httpMethod 参数为 HttpRequestBase 类型,HttpRequestBase 是一个抽象类,用于调用子类 HttpPost 实现类。

(3) 编写 packageParam()方法。

设置 HTTP 请求头向服务器发送请求,通过设置请求参数来指定获取哪些类型的数据内容,具体需要哪些参数以及这些参数的作用,会在后续的代码中进行讲解。在工具类中定义 packageParam()方法用于封装 HTTP 请求参数,该方法中包含两个参数,分别为 params 和 httpMethod,如文件 3-5 所示。

文件 3-5 HttpClientUtils.java

```
1 public static void packageParam(Map<String, String>params,
2     HttpEntityEnclosingRequestBase httpMethod)
3     throws UnsupportedEncodingException {
```

```
    //封装请求参数
    if (params !=null) {
        /**
         * NameValuePair 是简单名称值对节点类型。
         * 多用于 Java 向 url 发送 Post 请求。在发送
         * post 请求时用该 list 来存放参数。
         */
        List<NameValuePair>nvps =new ArrayList<NameValuePair>();
        /**
         * 通过 entrySet()方法从 params 中返回所有键值对的集合,
         * 并保存在 entrySet 中,通过 foreach 方法每次取出一
         * 个键值对保存在一个 entry 中。
         */
        Set<Entry<String, String>>entrySet =params.entrySet();
            for (Entry<String, String>entry : entrySet) {
                //分别提取 entry 中的 key 和 value 放入 nvps 数组中。
                nvps.add(new BasicNameValuePair(entry.getKey(),
                      entry.getValue()));
                }
            //设置到请求的 http 对象中,这里的 ENCODING 为之前创建的编码常量。
        httpMethod.setEntity(new UrlEncodedFormEntity(nvps,ENCODING));
    }
}
```

上述代码中,params 参数为 Map＜String,String＞类型,用于封装请求参数中的参数名称及参数值,httpMethod 参数为 HttpEntityEnclosingRequestBase 类型,是一个抽象类,其实现类包括 HttpPost、HttpPatch、HttpPut,是 HttpRequestBase 的子类,将设置的请求参数封装在 HttpEntityEnclosingRequestBase 对象中。

(4) 编写 HttpClientResp()方法。

前两步已经创建了封装请求头和请求参数的方法,按照 HTTP 请求的流程在服务器接收到请求后服务器将返回请求端响应内容,在工具类中定义 getHttpClientResult()方法用于获取 HTTP 响应内容,该方法中包含三个参数,分别为 httpResponse、httpClient 和 httpMethod,该方法包含返回值,返回值类型为之前定义的实体类 HttpClientResp,其中内容包括响应代码和响应内容,如文件 3-6 所示。

文件 3-6　HttpClientUtils.java

```
public static HttpClientResp
    getHttpClientResult(CloseableHttpResponse httpResponse,
    CloseableHttpClient httpClient, HttpRequestBase httpMethod)
    throws Exception {
    //通过请求参数 httpMethod 执行 HTTP 请求
    httpResponse =httpClient.execute(httpMethod);
    //获取 HTTP 的响应结果
    if (httpResponse !=null && httpResponse.getStatusLine() !=null){
        String content ="";
        if (httpResponse.getEntity() !=null) {
            //将响应结果转为 String 类型,并设置编码格式
```

```
            content =EntityUtils.toString(httpResponse.getEntity()
                        , ENCODING);
            }
        /**
        * 返回 HttpClientResp 实体类的对象,这两个参数分
        * 别代表实体类中的 code 属性和 content 属性,分别代
        * 表响应代码和响应内容。
        */
        return new HttpClientResp(httpResponse
                    .getStatusLine().getStatusCode(), content);
    }
    //如果没有接收到响应内容则返回响应的错误信息
    return new HttpClientResp(HttpStatus.SC_INTERNAL_SERVER_ERROR);
}
```

在上述代码中,创建的 getHttpClientResult()方法包含三个参数:httpResponse 参数为 CloseableHttpResponse 类型,用于在服务器接收并解释请求消息之后以 HTTP 响应消息进行响应,我们将要获取的响应内容就是通过该参数获取;httpClient 参数为 CloseableHttpClient 类型,用于表示 HTTP 请求执行的基础对象;httpMethod 参数为 HttpRequestBase 类型,用于实现 HttpPost。

(5) 编写 doPost()方法。

在前面创建了请求头、请求参数以及获取响应内容的方法。接下来将讲解通过 HttpClient Post 方式提交请求头和请求参数,从服务端返回状态码和 JSON 数据内容。(注意:选取请求方法要与爬取网站规定的请求方法一致,可以通过之前讲过的开发者模式中的 Network 查看,如图 3-12 所示),如文件 3-7 所示。

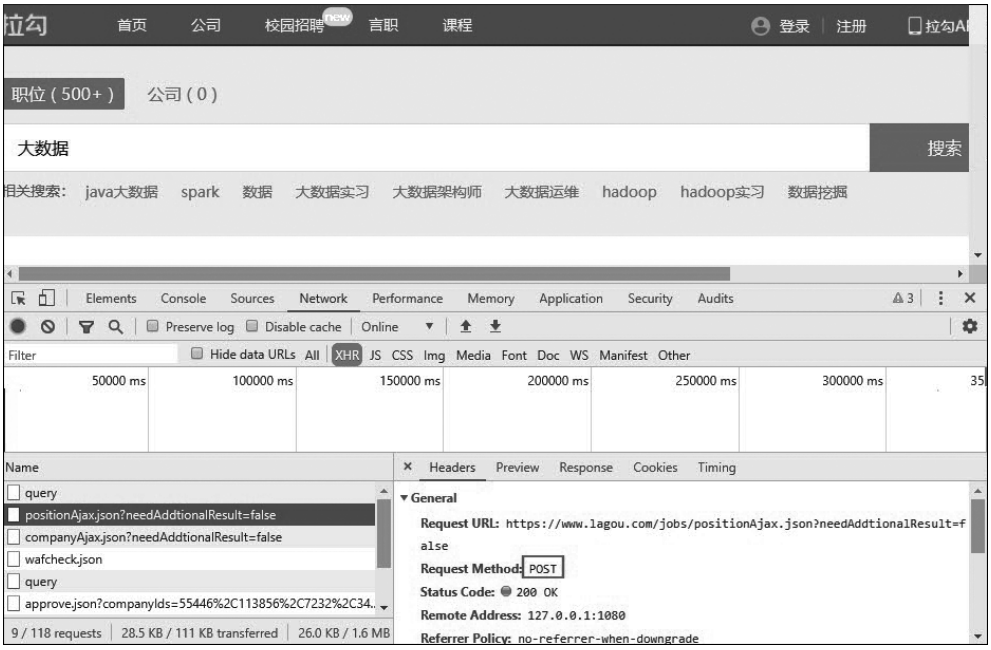

图 3-12 请求方法

文件 3-7　HttpClientUtils.java

```
1 public static HttpClientResp doPost(String url,
2     Map<String, String>headers,
3     Map<String, String>params) throws Exception {
4     //创建 httpClient 对象
5     CloseableHttpClient httpClient =HttpClients.createDefault();
6     //创建 httpPost 对象
7     HttpPost httpPost =new HttpPost(url);
8     /**
9      * setConnectTimeout:设置连接超时时间,单位毫秒。
10     * setConnectionRequestTimeout:设置从 connect Manager(连接池)
11     * 获取 Connection
12     * 超时时间,单位毫秒。这个属性是新加的属性,因为目前版本是可以共享连接池的。
13     * setSocketTimeout:请求获取数据的超时时间(即响应时间),单位毫秒。如果
14     * 访问一个接口,多少时间内无法返回数据,就直接放弃此次调用。
15     */
16    //封装请求配置项
17    RequestConfig requestConfig =RequestConfig.custom()
18            .setConnectTimeout(CONNECT_TIMEOUT)
19            .setSocketTimeout(SOCKET_TIMEOUT)
20            .build();
21    //设置 post 请求配置项
22    httpPost.setConfig(requestConfig);
23    //通过创建的 packageHeader()方法设置请求头
24    packageHeader(headers, httpPost);
25    //通过创建的 packageParam()方法设置请求参数
26    packageParam(params, httpPost);
27    //创建 httpResponse 对象获取响应内容
28    CloseableHttpResponse httpResponse =null;
29    try {
30        //执行请求并获得响应结果
31        return getHttpClientResult(httpResponse, httpClient, httpPost);
32    } finally {
33        //释放资源
34        release(httpResponse, httpClient);
35    }
36 }
```

上述代码中的 doPost()方法,定义的返回值类型为实体类 HttpClientResp 对象,方法中包含三个参数,分别如下。

① url：进行数据采集的网站链接。

② headers：请求头数据。

③ params：请求参数数据。

在 doPost()方法中调用已创建的 getHttpClientResult()方法获取响应结果数据并作为方法的返回值。

在释放资源一行代码会报错,因为释放资源方法需要通过自行创建后去调用,下面将编写释放资源的方法。

(6) 编写 release()方法。

HttpClient 在使用过程中要注意资源释放和超时处理的问题,如果线程资源无法释放,会导致线程一直在等待,最终导致内存或线程被大量占用。这里创建了一个释放资源的方法 release()主要用于释放 httpclient(HTTP 请求)对象资源和 httpResponse(HTTP 响应)对象资源,如文件 3-8 所示。

文件 3-8 HttpClientUtils.java

```
public static void release(CloseableHttpResponse httpResponse,
    CloseableHttpClient httpClient) throws IOException {
    //释放资源
    if (httpResponse !=null) {
        httpResponse.close();
    }
    if (httpClient !=null) {
        httpClient.close();
    }
}
```

至此,HttpClient 的所有工具类准备完毕,后续直接在实现网页数据采集的主类中调用这些方法即可,通过编写存储数据的工具类,实现将采集的网页数据存储到 HDFS 上。

3.3.3 封装存储在 HDFS 的工具类

通过前两节的操作可以成功采集招聘网站的数据,为了便于后续对数据的预处理和分析,需要将数据采集程序获取的数据存储到本地或者集群中的 HDFS 上,本节将详细讲解如何将爬取的数据存放到 HDFS 上。

(1) 在 pom.xml 文件中添加 Hadoop 的依赖,用于调用 HDFS API,代码如文件 3-9 所示。

文件 3-9 pom.xml

```
<dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>2.7.4</version>
</dependency>
<dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
        <version>2.7.4</version>
</dependency>
```

(2) 在 com.position.reptile 包下,创建名为 HttpClientHdfsUtils.java 文件的工具类,实现将数据写入 HDFS 的方法 createFileBySysTime(),该方法包括三个参数:url(表示 Hadoop 地址)、fileName(表示存储数据的文件名称)和 data(表示数据内容),如文件 3-10 所示。

文件 3-10　HttpClientHdfsUtils.java

```
1 public static void createFileBySysTime(String url,
2     String fileName,String data) {
3     //指定操作 HDFS 的用户
4     System.setProperty("HADOOP_USER_NAME", "root") ;
5     Path path =null;
6     //读取系统时间
7     Calendar calendar =Calendar.getInstance();
8     Date time =calendar.getTime();
9     //格式化系统时间为年月日的形式
10    SimpleDateFormat format =new SimpleDateFormat("yyyyMMdd");
11    //获取系统当前时间并将其转换为 string 类型,fileName 即存储数据的文件夹名称
12    String filePath =format.format(time);
13    //构造 Configuration 对象,配置 Hadoop 参数
14    Configuration conf =new Configuration();
15    //实例化 URI 引入 uri
16    URI uri =URI.create(url);
17    //实例化 FileSystem 对象,处理文件和目录相关的事务
18    FileSystem fileSystem;
19    try {
20        //获取文件系统对象
21        fileSystem =FileSystem.get(uri,conf);
22        //定义文件路径
23        path =new Path("/JobData/"+filePath);
24        //判断路径是否为空
25        if (!fileSystem.exists(path)) {
26            //创建目录
27            fileSystem.mkdirs(path);
28        }
29        //在指定目录下创建文件
30        FSDataOutputStream fsDataOutputStream =fileSystem.create(
31            new Path(path.toString()+"/"+fileName));
32        //向文件中写入数据
33        IOUtils.copyBytes(new ByteArrayInputStream(data.getBytes()),
34                fsDataOutputStream, conf, true);
35        //关闭连接释放资源
36        fileSystem.close();
37    } catch (IOException e) {
38        e.printStackTrace();
39    }
40 }
```

上述代码中,指定在 Hadoop 集群的 HDFS 上创建/JobData 目录,用于存储当天爬取的数据,数据将以文件的形式存储在由当前系统日期的“年月日”组成目录下。

至此,将数据存储到 HDFS 上的工具类准备完成,后续直接在实现网页数据采集的主类中调用该方法即可实现将采集的数据实时存储到 HDFS 上,通过编写主方法实现网页数据采集功能。

3.3.4 实现网页数据采集

实现网页数据采集的详细过程分为如下几个步骤。

(1) 获取网站的请求头内容,可通过 Chrome 浏览器进入开发者模式查看请求头的详细内容,如图 3-13 所示。

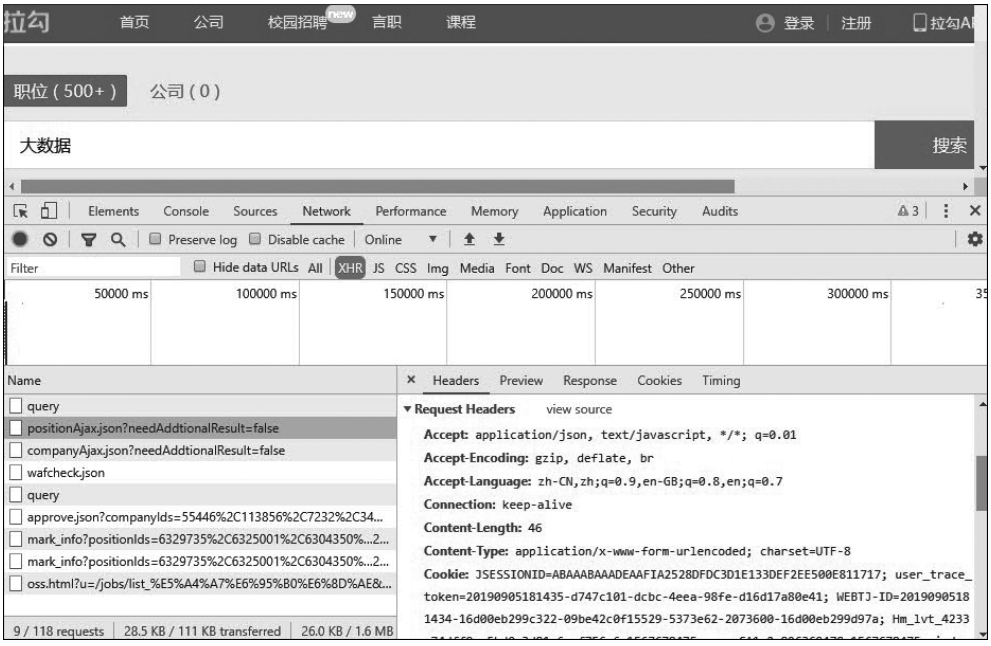

图 3-13 请求头数据

在图 3-13 中,将 Request Headers 一项中的参数以<Key,Value>形式写入到 Map 集合中作为数据采集程序的请求头,这么做的目的是模拟浏览器登录,Cookie 一栏参数只有登录才会产生,建议读者登录网站以获取 Cookie,防止爬虫失败。

(2) 在 com.position.reptile 包下,创建名为 HttpClientData.java 文件的主类,用于实现数据采集功能,在该类中创建 main()方法,在 main()方法中创建 Map 集合 headers,将请求头参数放入集合中,如文件 3-11 所示。

文件 3-11 HttpClientData.java

```
1 //设置请求头
2 Map<String, String>headers =new HashMap<String, String>();
3     headers.put("Cookie", "");
4     headers.put("Connection", "keep-alive");
5     headers.put("Accept",
6                 "application/json, text/javascript, */*; q=0.01");
7     headers.put("Accept-Language","zh-CN,zh;q=0.9,en-GB;q=0.8,en;q=0.7");
8     headers.put("User-Agent",
9                 "Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
10                +"AppleWebKit/537.36 (KHTML, like Gecko) "
11                +"Chrome/75.0.3770.142 Safari/537.36");
```

```
    headers.put("Content-Type",
                "application/x-www-form-urlencoded; charset=UTF-8");
    headers.put("Referer",
        "https://www.lagou.com/jobs/list_%E5%A4%A7%E6%95%B0%E6%8D%AE?"
        +"px=default&city=%E5%85%A8%E5%9B%BD");
    headers.put("Origin", "https://www.lagou.com");
    headers.put("X-Requested-With", "XMLHttpRequest");
    headers.put("X-Anit-Forge-Token", "None");
    headers.put("Cache-Control", "no-cache");
    headers.put("X-Anit-Forge-Code", "0");
    headers.put("Host", "www.lagou.com");
```

(3) 在 HttpClientData 类的 main()方法中再创建一个 Map 集合 params,将请求参数放入集合中,本项目主要使用三个参数来指定获取的数据类型,这三个参数包括:kd(职位类型)、city(城市)和 pn(页数)。其中,pn 参数需要在每次 HTTP 请求中发生递增变化,作用在于爬取不同页面中的数据,因此向集合 params 中添加 pn 参数的操作应放在循环中进行,下面通过编写代码设置请求参数,如文件 3-12 所示。

文件 3-12　HttpClientData.java

```
Map<String, String>params =new HashMap<String, String>();
params.put("kd", "大数据");
params.put("city", "全国");
for (int i =1; i <31; i++) {
    params.put("pn", String.valueOf(i));
}
```

通过上述代码中指定的请求参数可以看出,本实训项目获取的职位数据为全国的大数据相关职位信息,获取 30 页的数据内容进行后续的分析工作。

注意:参数名称要与网站指定的参数名称一致,可通过浏览器进入开发者模式进行查看,如果参数错误会导致客户端发送的请求服务端无法解析,无法获取数据。

(4) 请求头和请求参数设置完毕后,通过 HttpClient 的 post 请求实现数据的获取,因为需要获取不同页面的数据,每个页面的 pn 参数都会发生变化(请求参数发生变化),因此获取数据的方法要放在上一步实现的 for 循环中,通过变化的参数获取数据,将数据保存到 HDFS 上,更新后的 for 循环内部代码如文件 3-13 所示。

文件 3-13　HttpClientData.java

```
for (int i =1; i <31; i++) {
    params.put("pn", String.valueOf(i));
    HttpClientResp result =HttpClientUtils
                .doPost("https://www.lagou.com/jobs/positionAjax.json?"
                +"needAddtionalResult=false&first=true&px=default",
                headers,params);
    HttpClientHdfsUtils.createFileBySysTime("hdfs://hadoop01:9000",
                    "page"+i,result.toString());
    Thread.sleep(1 * 500);
}
```

在上述代码中，第 7～8 行代码调用 HDFS 工具类中的 createFileBySysTime()方法，该方法所需要的三个参数分别为：Hadoop 地址、文件名和数据内容，用于实现将每一页的数据以文件的形式存储到 HDFS 上；第 9 行代码设置请求的间隔时间，便于防止请求过快而被服务器屏蔽。

在 Eclipse 开发工具中运行主类文件 HttpClientData.java，最终将采集的数据存储到 HDFS 的/JobData/20190807 中，程序运行完成后在三台虚拟机中任意一台执行 Shell 指令“hdfs dfs -ls /JobData/20190807”，均可查看最终采集的数据结果，需要注意的是 hdfs 上的创建目录名称是根据运行程序的时间而定，所以需根据个人运行程序的时间对指令中的目录进行修改，最终采集的数据结果如图 3-14 所示。

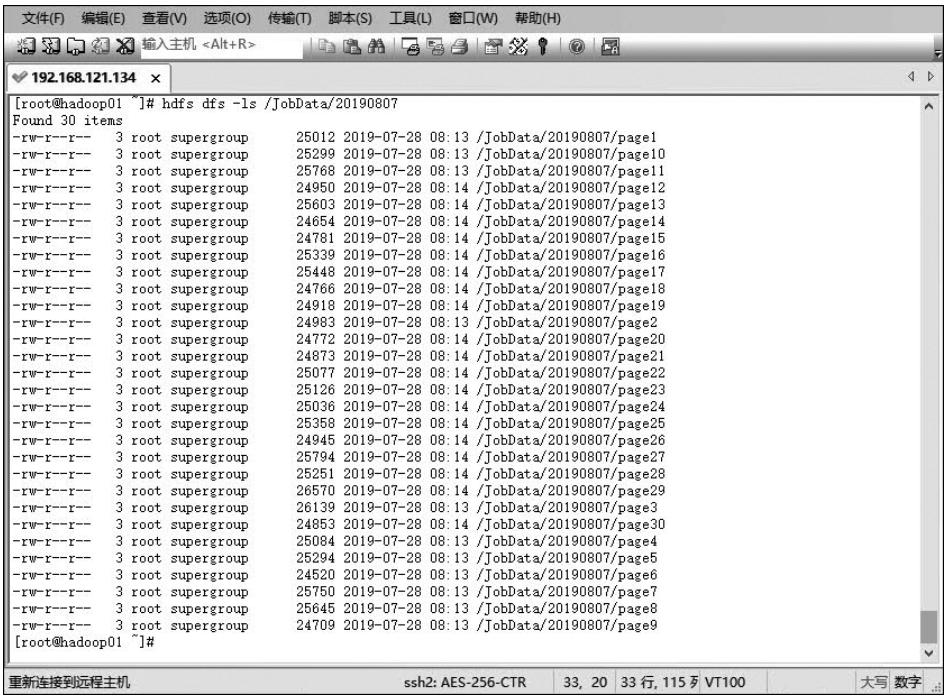

图 3-14　数据结果

注意：如读者在爬取数据时遇到问题，例如 IP 或者用户被锁定，导致数据无法获取，可在网页中退出当前登录的账户并清除浏览器缓存后关闭浏览器，等候几分钟后再次登录账户获取 Cookie，为了避免类似情况的发生，应避免频繁地爬取数据。因为爬取数据存在不可控性，若无法获取数据，读者也可在本书提供的配套资源中下载使用已经准备好的数据。

小结

本章主要讲解网络数据采集程序的编写。首先，通过理论基础方面了解数据采集相关知识内容，其中包括数据源的分类、HTTP 的请求过程和 HttpClient 框架的基本介绍。然后，通过分析采集数据的结构制定程序的编写方案以及编写采集程序环境的准备。最后，实际开发一个爬取招聘网站数据的程序。通过本章的学习，读者可掌握通过 HttpClient 框架进行爬虫的技巧，熟悉编写爬虫程序的操作流程。