第 3 章

JSON 数据爬取

JSON 是一种轻量级的数据交换格式。在了解 JSON 之前先来了解一下 Ajax 技术。

3.1 Ajax

3.1.1 Ajax 技术

有时在用 Requests 爬取网页时，得到的结果可能和在浏览器中看到的不一样。在浏览器中正常显示的网页数据，但是对网页 URL 发送请求爬取的结果并不包括想要的数据，如下面这个例子。

【例 3-1】 爬取腾讯技术类招聘首页所有招聘岗位的基本信息，爬取目标如图 3.1 所示。

图 3.1 截取腾讯技术类招聘当前页面

【解析】 操作步骤如下，对应如图 3.2 所示。

第①步：打开 Network 面板。

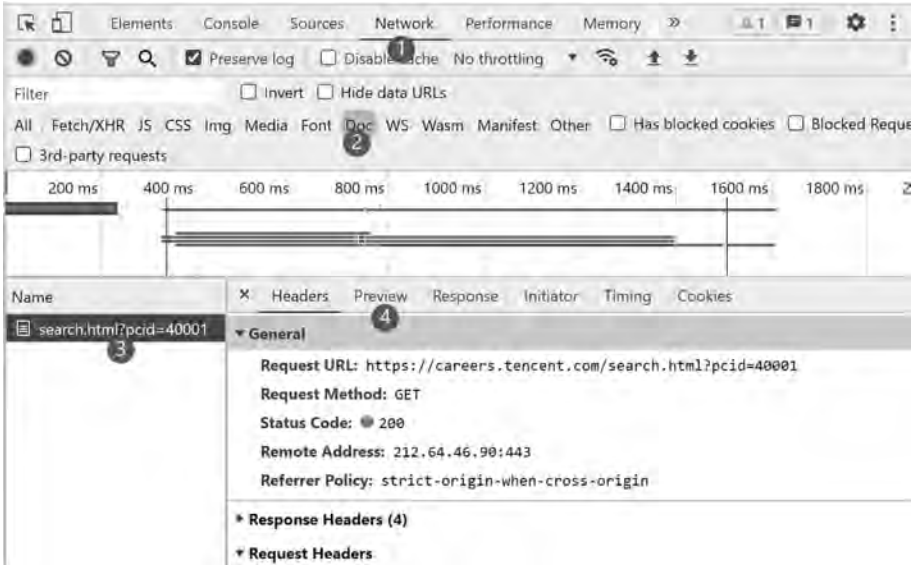

图 3.2　查看首页的 Headers 信息

第②步：按照第 2 章所讲的案例，单击 Doc 按钮，在请求列表中只有一个 Doc 类型的文件。

第③步：单击请求资源列表中的文件。

第④步：单击查看对应的 Preview，预览发现并没有对应的招聘信息，如图 3.3 所示。

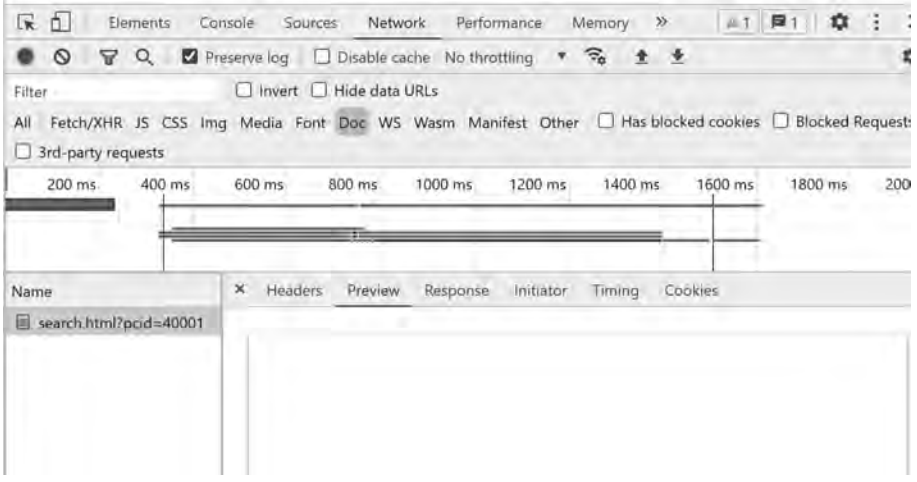

图 3.3　查看 Preview 信息

这是因为此网页中的各种招聘信息是经过 JavaScript 处理后生成的数据，这些数据的来源有多种，可能是通过 Ajax 加载的，也可能是包含在 HTML 文档中的，还有可能是经过 JavaScript 和特定算法计算后生成的。

对于第一种情况，数据加载是一种异步加载方式，原始的网页最初不会包含某些数据，原始网页加载完后，会再向服务器请求某个接口获取数据，然后数据才被处理，从而呈现到

网页上，这就是发送了一个 Ajax 请求。随着 Web 的发展，这种形式的网页越来越多。网页的原始 HTML 文档不会包含任何数据，数据都是通过 Ajax 统一加载后再呈现出来的，这样在 Web 开发上可以做到前后端分离，而且降低服务器直接渲染网页带来的压力。所以，如果遇到这样的网页，直接利用 Requests 库来爬取原始网页是无法获取到需要的数据的。

Ajax，全称为 Asynchronous JavaScript and XML，即异步的 JavaScript 和 XML。它不是一门编程语言，而是利用 JavaScript 在保证网页不被刷新、网页链接不改变的情况下与服务器交换数据并更新部分网页的技术。

对于传统的网页，如果需要更新其内容，那么必须要刷新整个网页，但有了 Ajax，便可以在网页不被全部刷新的情况下更新其内容。在这个过程中，网页实际上是在后台与服务器进行了数据交互，爬取数据之后，再利用 JavaScript 改变网页，这样网页内容就更新了。简单地说，在用户浏览网页的同时，局部更新网页中一部分数据。Ajax 提高用户浏览网站应用的体验感。

下面简单了解一下，从发送 Ajax 请求到网页更新这一网页内容加载过程的操作步骤，可以分为 3 步。

1. 发送请求

JavaScript 可以实现网页的各种交互功能，Ajax 也不例外，它也是由 JavaScript 实现的，如图 3.4 所示。

```
var xmlhttp;
if (window.XMLHttpRequest) {①
   xmlhttp=new XMLHttpRequest();
} else {
   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function() {②
   if (xmlhttp.readyState==4 && xmlhttp.status==200) {
      document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
   }
}
xmlhttp.open("POST","/ajax/",true);③
xmlhttp.send();
```

图 3.4　Ajax 代码

这是 JavaScript 对 Ajax 最底层的实现。

第①步：新建了 XMLHttpRequest 对象。

第②步：调用 onreadystatechange 属性设置了监听。

第③步：调用 open()和 send()方法向服务器发送了请求。

由 JavaScript 来完成发送请求，由于设置了 onreadystatechange 监听，所以当服务器返回响应时，onreadystatechange 对应的方法便会被触发，然后在这个方法里解析响应内容。

2. 解析当前页内容

得到响应之后，onreadystatechange 属性对应的方法便会被触发，此时利用 xmlhttp 的 responseText 属性便可爬取到响应内容。这类似于 Python 中利用 Requests 库向服务器发起请求，然后得到响应的过程。那么返回内容可能是 HTML，也可能是 JSON，接下来只需要在方法中用 JavaScript 进一步处理即可。例如，如果是 JSON 的话，可以进行解析和转化。

3. 渲染网页

JavaScript 有改变网页内容的能力，解析完响应内容之后，就可以调用 JavaScript 来针对解析完的内容对网页进行下一步处理。例如通过 document.getElementById(). innerHTML 这样的操作，便可以对某个元素内的源代码进行更改，网页显示的内容跟着改变，这样的操作也被称作 DOM 操作，即对 Document 网页文档进行操作，如更改、删除等。

前面 document.getElementById("myDiv").innerHTML＝xmlhttp.responseText 是将 ID 为 myDiv 的节点内部的 HTML 代码更改为服务器返回的内容，myDiv 元素内部便会呈现出服务器返回的新数据，网页的部分内容就更新了。

那么 Ajax 异步动态加载的数据在爬虫时应该如何爬取？

3.1.2 分析数据来源

以 Chrome 浏览器为例，分析例 3-1 中腾讯招聘官网上的技术类招聘职位数据来源。

URL 地址为 https://careers.tencent.com/search.html?pcid＝40001。操作步骤如下，对应如图 3.5 所示。

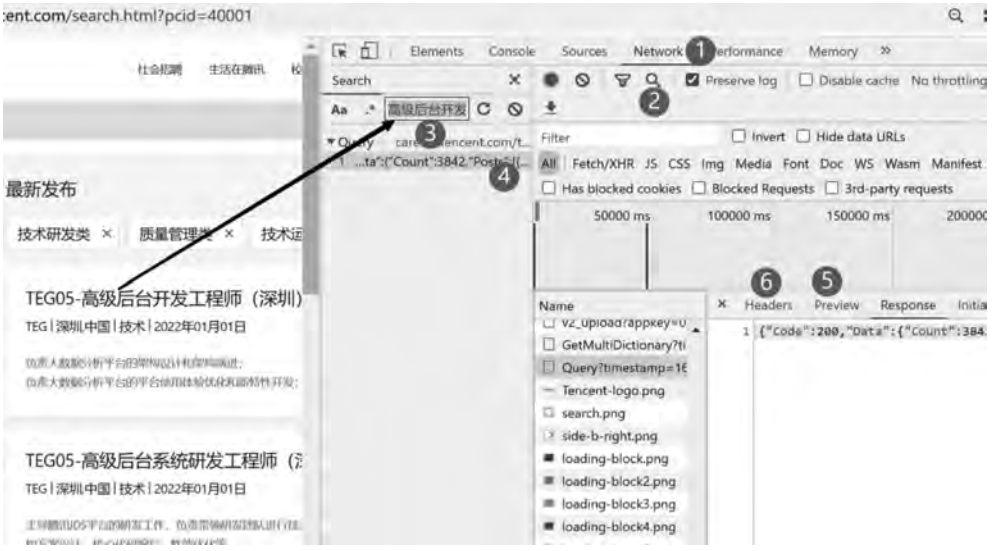

图 3.5 分析数据源解析流程

第①步：打开 Network 面板。

第②步：单击控制器上的搜索按钮，出现了一个搜索栏。

第③步：在搜索框输入需要爬取数据内容的任意某几个字，如输入“高级后台开发”。

第④步：在搜索得到的结果中单击最里层数据，请求资源列表栏会自动出现对应的 Response 面板，从此面板里可以查看数据是否为需要的数据。

第⑤步：单击 Preview 标签，查看响应资源数据的预览信息，如图 3.6 所示。

第⑥步：单击 Headers 标签，查看响应资源数据的 Headers 信息，如图 3.7 所示，找到请求地址 Request URL、请求方法 Request Method 和查询参数 Query string Parameters，为写爬虫程序做准备。

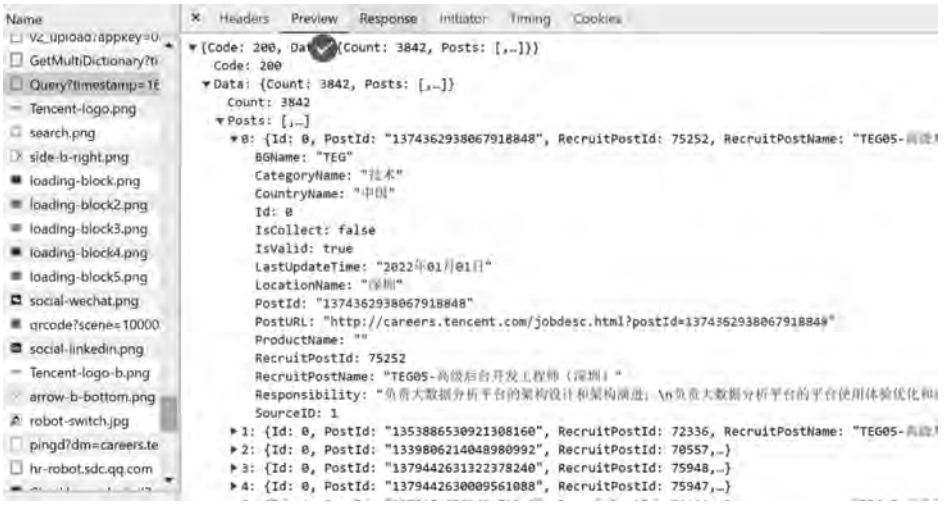

图 3.6　查看 Preview 信息

图 3.7　查看 Headers 信息

代码如下：

```
import requests
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "https://careers.tencent.com/tencentcareer/api/post/Query"
keys = {
    'timestamp':'1641078092667',
      'countryId':'',
      'cityId': '',
      'bgIds': '',
      'productId': '',
      'categoryId': '',
      'parentCategoryId': '40001',
      'attrId': '',
```

```
        'keyword':'',
        'pageIndex': '1',
        'pageSize': '10',
        'language': 'zh-cn',
        'area': 'cn'
}
response = requests.get(url = url,headers = headers,params = keys)
response.json()
```

补充说明，查看 Headers 面板，发现参数栏里查询参数有 13 个参数，如果不确定哪些有用，哪些没用，直接全部以键值对的形式存在字典参数中，空值赋值为空字符串就可以。

程序执行的结果如图 3.8 所示。

{'Code': 200,
 'Data': {'Count': 3842,
  'Posts': [{'Id': 0,
    'PostId': '1374362938067918848',
    'RecruitPostId': 75252,
    'RecruitPostName': 'TEG05-高级后台开发工程师（深圳）',
    'CountryName': '中国',
    'LocationName': '深圳',
    'BGName': 'TEG',
    'ProductName': '',
    'CategoryName': '技术',
    'Responsibility': '负责大数据分析平台的架构设计和架构演进；\n负责大数据分析平台的平台使用体验优化和新特性开发；\n负责大数据分析平台各
模块性能优化和日常运维\n',
    'LastUpdateTime': '2022年01月01日',
    'PostURL': 'http://careers.tencent.com/jobdesc.html?postId=1374362938067918848',
    'SourceID': 1,
    'IsCollect': False

图 3.8　执行结果的部分截图

通过查看程序执行结果可以得知，这是一个 JSON 字符串，为了帮助实现后续的数据分析，下面介绍一下 JSON 数据的基本语法。

3.2　JSON

JSON(JavaScript Object Notation)是一种轻量级的数据交换格式。它使得人们很容易地进行阅读和编写，同时也方便机器进行解析和生成。它是基于 JavaScript Programming Language，Standard ECMA-262 3rd Edition-December 1999 的一个子集。JSON 采用完全独立于程序语言的文本格式，使用类似 C 语言的习惯（包括 C、C++、C＃、Java、JavaScript、Perl、Python 等），这些特性使 JSON 成为理想的数据交换语言。JSON 主要基于两种结构：

（1）“名称/值”对的集合。

“名称/值”对在不同的编程语言中，分别被理解为对象、记录、结构、字典、哈希表、有键列表，或者关联数组等。

（2）值的有序列表。

在大部分语言中，值的有序列表被实现为数组、矢量、列表、序列等。

以上这些都是常见的数据结构。目前，绝大部分编程语言都以某种形式支持它们，这使得在各种编程语言之间交换同样格式的数据成为可能。

3.2.1　JSON 语法规则

JSON 具有以下形式。

1. 值

值 value 可以是双引号括起来的字符串 string、数值 number、逻辑值 true 或 false、null、对象 object 或者数组 array 等，并且这些结构可以嵌套，如图 3.9 所示。

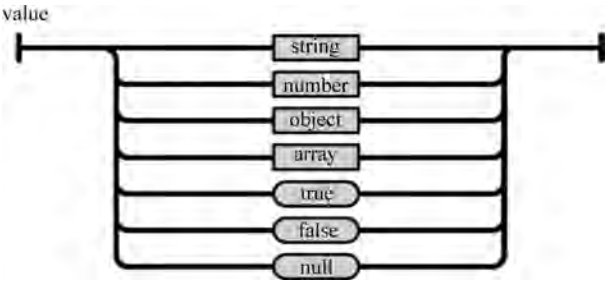

图 3.9　JSON 值语法规则

2. 字符串

字符串 string 是由双引号包围的任意数量 Unicode 字符的集合，使用反斜线转义。一个字符 character 即一个单独的字符串 character string。JSON 的字符串 string 与 C 或者 Java 的字符串非常相似，如图 3.10 所示。

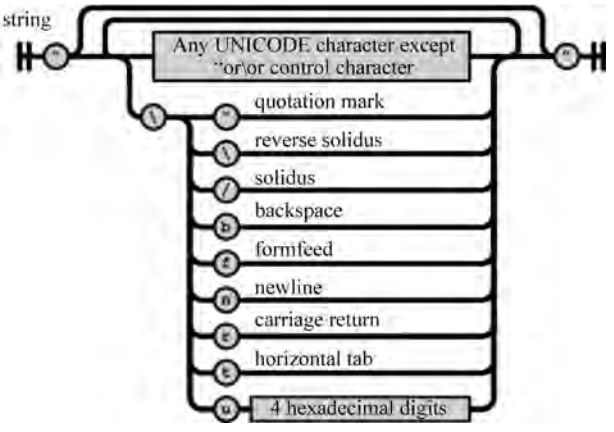

图 3.10　JSON 字符串语法规则

3. 数值

数值 number，也与 C 或者 Java 的数值非常相似，如图 3.11 所示。只是 JSON 的数值没有八进制与十六进制格式。

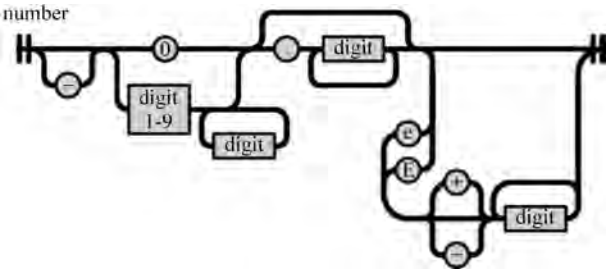

图 3.11　JSON 数值语法规则

4. 对象

对象 object 是一个无序的“名称/值”对集合。一个对象以左括号“{”开始，以右括号“}”结束。每个“名称”后跟一个冒号“:”，“名称/值”对之间使用逗号“,”分隔，如图 3.12 所示。

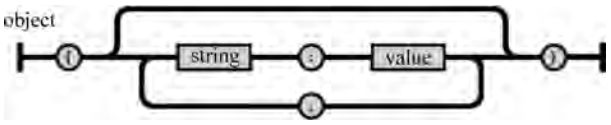

图 3.12 JSON 对象语法规则

【例 3-2】 一个 JSON 对象实例。

```
{
    "name":"Jack",
    "at large": true,
    "grade": "A",
    "format": {
        "type":"rect",
        "width": 1920,
        "height":1080,
        "interlace": false,
        "framerate": 24
    }
}
```

5. 数组

数组 array 是值 value 的有序集合。一个数组以左中括号“[”开始，以右中括号“]”结束。值之间使用逗号“,”分隔，如图 3.13 所示。

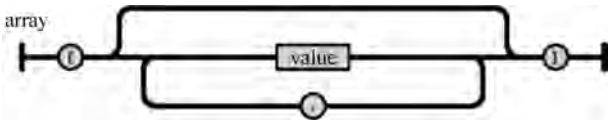

图 3.13 JSON 数组语法规则

【例 3-3】 一个包含三个对象的 JSON 对象。

```
{
    "name": "JSON 中国",
    "url": "http://www.json.org.cn",
    "links": [
        {"name": "Google","url": "http://www.google.com" },
        { "name": "Baidu","url": "http://www.baidu.com" },
        { "name": "SoSo", "url": "http://www.SoSo.com" } ]
}
```

3.2.2 访问 JSON 数据

JSON 语法格式中，对象类似于 Python 的字典，数组类似于 Python 的列表，通过索引访问数组中的对象，索引从 0 开始。

【例 3-4】 访问下面这个 JSON 对象中的某个值，如取出“links”中“http://www.SoSo.com”值。

```
infor = {
    "name": "JSON 中国",
    "url": "http://www.json.org.cn",
    "links": [
        {"name": "Google","url": "http://www.google.com" },
        { "name": "Baidu","url": "http://www.baidu.com" },
        { "name": "SoSo", "url": "http://www.SoSo.com" } ]
    }
```

【解析】 JSON 对象类似于 Python 的字典，通过键可以取出值。JSON 数组类似 Python 列表，通过索引定位，索引从 0 开始。

```
infor['links']
```

输出值：

```
[{'name': 'Google', 'url': 'http://www.google.com'},
 {'name': 'Baidu', 'url': 'http://www.baidu.com'},
 {'name': 'SoSo', 'url': 'http://www.SoSo.com'}]
infor['links'][2]
```

输出值：

```
{'name': 'SoSo', 'url': 'http://www.SoSo.com'}
infor['links'][2]['url']
```

输出值：

```
'http://www.SoSo.com'
```

3.2.3 JSON 文件读写操作

JSON 的文件类型后缀是.json，爬取下来的 JSON 数据以.json 格式保存，在 Python 中使用 json.dump()和 json.load()实现 JSON 文件的读写操作。

【例 3-5】 把一个名为“infor”的 JSON 对象存储为文件。

```
import json
infor = {
    "name": "JSON 中国",
    "url": "http://www.json.org.cn",
    "links":
        [{"name": "Google","url": "http://www.google.com" },
         { "name": "Baidu","url": "http://www.baidu.com" },
         { "name": "SoSo", "url": "http://www.SoSo.com" } ]
        }
with open('jsonfile.json', 'w') as fp:
    json.dump(infor, fp)
```

执行完程序，打开本地 jsonfile.json 文件，查看结果如图 3.14 所示。

图 3.14　jsonfile.json 信息

【例 3-6】 读取例 3-5 得到的 jsonfile.json 文件数据，并打印输出。

```
import json
with open('jsonfile.json') as file_obj:
    numbers = json.load(file_obj)
    print(numbers)
```

程序执行结果如图 3.15 所示。

```
{'name': 'JSON中国', 'url': 'http://www.json.org.cn', 'links': [{'name': 'Google', 'ur
l': 'http://www.google.com'}, {'name': 'Baidu', 'url': 'http://www.baidu.com'}, {'nam
e': 'SoSo', 'url': 'http://www.SoSo.com'}]}
```

图 3.15　JSON 文件数据读取结果

3.2.4　JSON 数据校验和格式化

不论是从例 3-5 中写入文档中的 JSON 数据，还是从文档中读出来的 JSON 数据，数据之间的层级关系都不够清晰。目前有很多网站提供 JSON 数据在线编辑工具，把层级不清晰的 JSON 数据格式化为清晰的层级关系。

例如网站 http://www.json.org.cn/tools/JSONLint/index.htm 提供了 JSON 校验和格式化功能。把例 3-5 得到的 jsonfile.json 文档中的数据复制到此网站编辑区域，单击格式化之后效果如图 3.16 所示，展现出了清晰的数据层级关系。

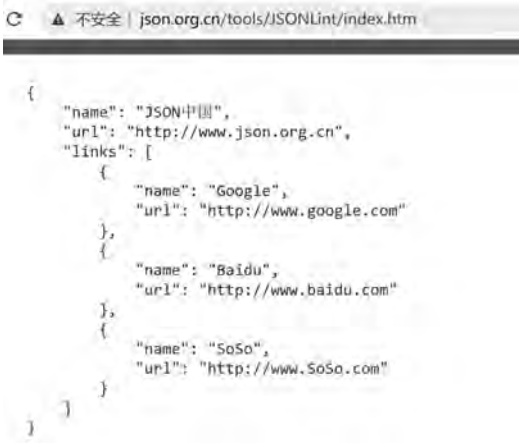

图 3.16　JSON 在线格式化结果

3.3 Ajax 异步动态加载的数据爬虫

3.3.1 带参数的 POST 请求爬虫

【例 3-7】 爬取网站 https://www.bjotc.cn/listing/list2.html?key=113,-1 中各公司名称和相关企业介绍信息，爬取对象如图 3.17 所示。

图 3.17 爬取对象

爬取这些公司数据，首先要找到数据所在的资源文件，然后对资源文件发送请求。操作步骤如下，对应如图 3.18 所示。

第①步：打开 Network 面板。

第②步：在控制器工具栏单击搜索工具。

第③步：在出现的搜索框中输入要爬取的数据中的任意几个字。

第④步：单击对应反馈的搜索结果。

第⑤步：请求列表自动停留在 Response 面板。

第⑥步：单击 Preview 标签，预览数据，确认数据是否正确，如图 3.19 所示。

第⑦步：单击 Headers 标签，查看该资源文件 Headers 信息，如图 3.20 所示，找到 Request URL，Request Method，Form Data，为写爬虫程序作准备。

通过查看头部信息可知，这是一个带参数的 POST 请求，POST 请求方法如下所示：

```
response = requests.post(url = url, headers = headers, data = keys)
```

其中形参名为 data。

图 3.18　查找数据所在的资源文件

图 3.19　查看预览信息

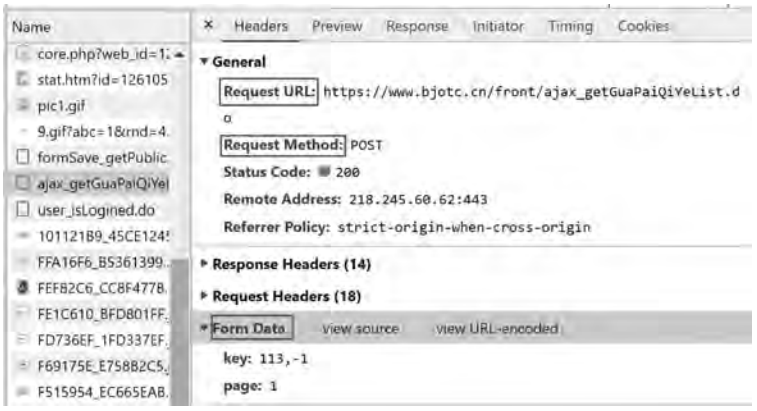

图 3.20　查看头部信息

代码如下：

```
import requests
import json
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "https://www.bjotc.cn/front/ajax_getGuaPaiQiYeList.do"
keys = {
    "key": "113, - 1",
    "page": "1"
}
response = requests.post(url = url,headers = headers,data = keys)
response.text
```

程序执行结果如图 3.21 所示。

图 3.21　爬取数据结果部分截图

3.3.2　多个网页多链接 GET 请求爬虫综合案例

【例 3-8】 爬取腾讯网站上各职业所有招聘岗位的详细数据，包括名称、地址、类别、时间、工作职责、工作要求等数据，爬取目标如图 3.22 所示。

最终需要爬取的目标数据包括技术类，产品类，内容类，设计类，销售、服务与支持类，人

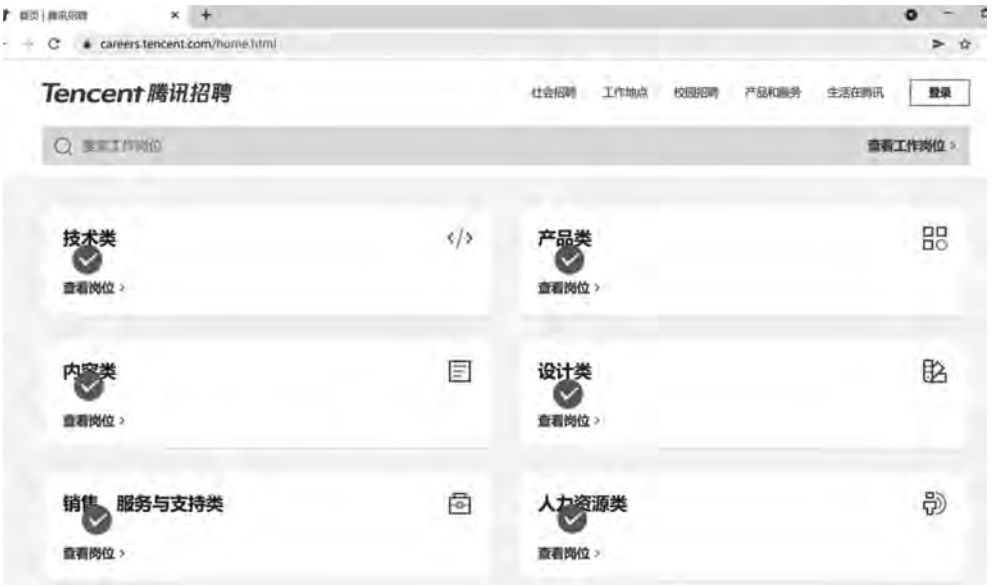

图 3.22　各招聘分类

力资源类共 6 类；在 Web 上打开某一职业类进入新的网页，可以看到这一类职业的多个招聘岗位；打开每个岗位链接，再次进入新的网页爬取具体岗位的详细招聘数据，爬取目标如图 3.23 所示。所以这个爬虫过程需要多次翻页操作，直到找到具体的每一个岗位的招聘信息，收集到数据之后返回上一页，去收集下一页数据，所有页收集完成之后返回上一类。

图 3.23　数据的翻页示意

主要解题思路分析如下：

第①步：查看如图 3.23 所示的每一个页面的 Network 面板，可以发现招聘数据不是静态网页，是实时动态加载的。

第②步：因为是动态加载的数据，通过查看可知请求得到的资源数据是 JSON 格式。

第③步：从 JSON 文件中解析出每个职业类的类 ID，以及每个岗位的 PostId。

第④步：PostId 作为参数控制进入每个岗位，从而获取每个岗位的详细数据。

第⑤步：整个操作从内到外进行，首先爬取某一个招聘岗位的具体信息，然后爬取某一页所有招聘岗位的具体信息，然后爬取某一类所有页所有招聘岗位的具体信息，最后实现爬取多类所有页所有招聘岗位的具体信息。

下面从内到外逐步实现上述操作过程。

1. 单个招聘岗位的详细数据爬取

这一步主要是实现爬取某一个招聘岗位的详细信息，包括名称、地址、类别、时间、工作职责、工作要求数据，并存到 CSV 文件中，如图 3.24 所示。

图 3.24　单个岗位需要爬取的数据

操作步骤如下，对应如图 3.25 所示。

第①步：打开 Network 面板。

第②步：在控制器工具栏单击搜索工具。

第③步：在出现的搜索框，输入要爬取的数据中的几个字。

第④步：单击下面反馈的搜索结果。

第⑤步：请求列表资源自动被选中，而且默认停留在 Response 面板，该响应的数据是一个 JSON 数据。

第⑥步：单击 Preview，预览数据，确认是否为想要的数据，如图 3.26 所示。

第⑦步：查看该资源文件的头部信息，如图 3.27 所示，找到 Request URL、Request Method、Form Data，为写爬虫程序作准备。

代码实现如下：

```
import requests
```

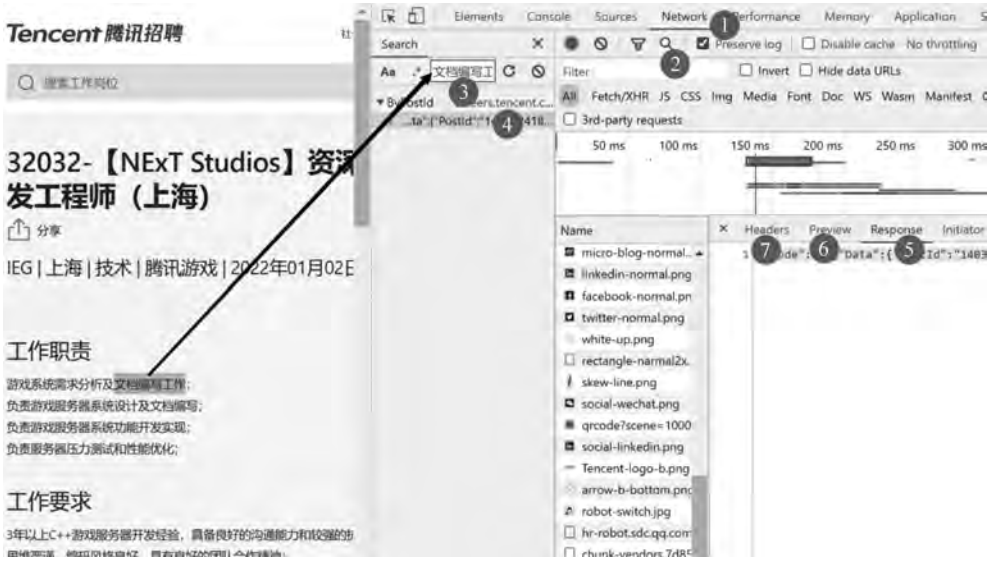

图 3.25　定位资源数据

图 3.26　预览数据

```
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "https://careers.tencent.com/tencentcareer/api/post/ByPostId"
keys = {
     'timestamp': '1638020073688',
    'postId': '1422487673381068800',
    'language': 'zh - cn'
}
response = requests.get(url = url,headers = headers,params = keys)
response.json()
```

程序执行结果如图 3.28 所示。

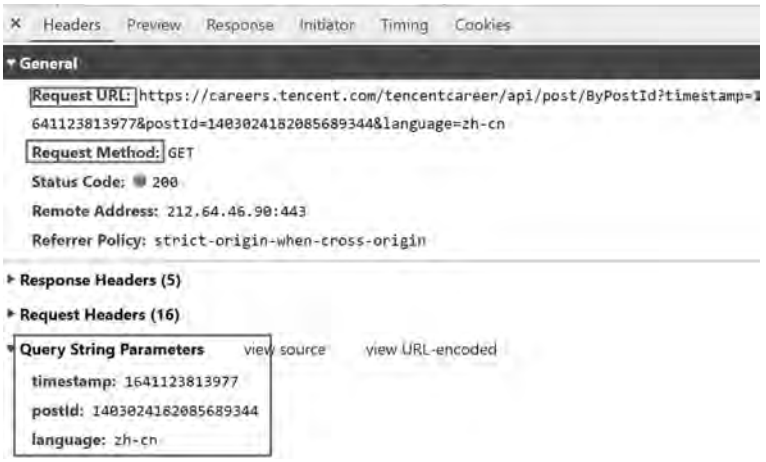

图 3.27　查看头部信息

图 3.28　一个岗位的招聘信息

爬取的 JSON 数据，可以直接保存成后缀为.json 的类型文件，也可以把 JSON 数据解析和提取，并通过 Pandas 保存成后缀为.csv 的文件，代码如下：

```
import pandas as pd
infor = response.json()
row = [infor['Data']['RecruitPostName'], infor['Data']['LocationName'], infor['Data']
['CategoryName'], infor['Data']['LastUpdateTime'], infor['Data']['Responsibility'], infor['Data']
['Requirement']]
headers = ['岗位','地址','类别','时间','职责','要求']
dict_infor = dict(zip(headers,row))
dataframe = pd.DataFrame(dict_infor, index = [0])
dataframe.to_csv('position.csv', mode = 'a', index = False, sep = ',', header = False)
```

执行程序，打开保存成后缀为.csv 的文件，结果如图 3.29 所示。

2. 单个网页多链接数据爬取

上一步实现了单个岗位详细信息的爬取，下面要爬取上一个网页中所有岗位的详细数据信息。爬取目标如图 3.30 所示。

18436-实时计算高级开发工程师	深圳	技术	2021年12月27日	1、参与数仓流批一体架构设计，建设新型实时数仓，强化数据服务能力 2、参与实时平台技术架构设计、质量把控、性能优化及难点攻克 3、参与大数据技术体系&大数据平台的规划，打造业界领先的实时计算开发平台	1、计算机或相关专业本科及以上学历，大数据方向5年以上相关工作经验 2、熟悉大数据生态组件，包括Hadoop、Hive、Spark、Hbase、Flink、Kafka等，对一种或多种有深入的理解 3、熟练使用Flink，并且对Flink的底层原理有很深的理解 4、熟练使用Java，Python，Scala语言中一种或者多种，有大数据量、高并发处理经验，有处理上亿用户数据的经验优先

图 3.29　保存成后缀为.csv 的文件内容

图 3.30　所有岗位的爬取目标

打开每个待爬取岗位的 Network 面板，查看要爬取的头部 Headers 面板，发现每个详情页有一个共性：Request URL 和 Request Method 方法相同，而参数 Form Data 不同，如图 3.31 和图 3.32 所示。

【说明】 “timestamp”是时间戳参数，由系统自动生成，这个参数可以直接复制过来。

既然 Request URL 和 Request Method 方法相同，参数不同，那么就可以使用同一个爬虫程序，只需要传递不同 PostId 参数就可以。参数不同，爬取的数据就是不同招聘岗位的具体数据。那么如何找到每个岗位的 PostId 呢？通过分析，可以发现这个参数在岗位浏览页信息中，如图 3.33 所示。此网页中数据仍然是动态加载 JSON 数据，定位数据资源的方法和前面相同，这里不再赘述，只需要观察一下数据的特点。

定位到要爬取的数据，在预览信息里可查看数据是 JSON 格式，并且里面包括了每个岗位的 PostId，只需要从 JSON 数据里解析出 PostId，把它作为参数传给上一步程序就可以了。下面，来爬取招聘岗位浏览页的 JSON 数据，首先去查看该招聘岗位浏览页所对应 Headers 信息，如图 3.34 所示。

代码如下：

图 3.31 “互娱流量接入系统测试开发工程师岗位数据”响应资源头部 Headers 面板

图 3.32 “IEG 增长中台电竞生态技术总监”响应资源头部 Headers 面板

```
import requests
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36'}
url = "https://careers.tencent.com/tencentcareer/api/post/Query"
keys = {
    'timestamp': '1638023920104',
```

图 3.33　查看预览 Preview 面板

图 3.34　查看岗位浏览页所对应的头部信息

```
    'countryId': '',
    'cityId': '',
    'bgIds': '',
    'productId': '',
    'categoryId': '',
    'parentCategoryId': '40001',
    'attrId': '',
    'keyword': '',
    'pageIndex': '1',
    'pageSize': '10',
    'language': 'zh-cn',
    'area': 'cn'
}
response1 = requests.get(url = url,headers = headers,params = keys)
postinfor = response1.json()
for i in postinfor['Data']['Posts']:
        print(i['PostId'])
```

代码执行的结果如下：

```
1403024182085689344
1351128060597903360
1401892346097836032
1370254025882083328
1446372730491379712
1422401488839254016
1409423315281387520
1435525814299926528
1424556800878845952
1244541464873013248
```

成功地爬取到每个招聘岗位的 PostId，把 PostId 存放在列表中，循环调用上一步的爬取程序，就能获得所有招聘岗位的详细数据信息。

进一步修改代码如下：

```
import requests
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36'}
def one_post_infor(PostId): #爬取岗位为 PostId 的数据信息
    url = "https://careers.tencent.com/tencentcareer/api/post/ByPostId"
    keys = {
        'timestamp': '1638020073688',
        'postId':PostId ,#可变参数
        'language': 'zh-cn'
    }
    response = requests.get(url = url,headers = headers,params = keys)
    infor = response.json() row = [infor['Data']['RecruitPostName'], infor['Data']['LocationName'],infor['Data']['CategoryName'],infor['Data']['LastUpdateTime'],infor['Data']['Responsibility'],infor['Data']['Requirement']]
    csv_headers = ['岗位','地址','类别','时间','职责','要求']
    dict_infor = dict(zip(csv_headers,row))
    dataframe = pd.DataFrame(dict_infor,index = [0])
    dataframe.to_csv('position.csv',mode = 'a',index = False,sep = ',',header = False)

for i in postinfor['Data']['Posts']:
        one_post_infor(i['PostId'])
```

程序执行的结果如图 3.35 所示，爬取网页中 10 个岗位的基本信息，保存成后缀为.csv 的文件。

	A	B	C	D	E	F	G
1	32032-【NExT Studios】资深后台开发工	上海	技术	2022年1月2日	游戏系统需3年以上C++游戏服务器		
2	23295-互娱流量接入系统测试开发工	深圳	技术	2022年1月2日	1、参与系统	1、计算机或相关专业本	
3	23295-IEG增长中台电竞生态技术总	深圳	技术	2022年1月2日	1、负责腾讯	1、5年以上工作经验，	
4	25927-移动游戏高级安全工程师（成	成都	技术	2022年1月2日	负责移动游戏	本科以上学历，3年以上	
5	18428-金融科技财付通创新业务研发	深圳	技术	2022年1月2日	1、负责金融	1、计算机、通信相关专	
6	36242-数据开发工程师（深圳）	深圳	技术	2022年1月2日	1、负责风险	1、具备实际的大数据业	
7	CSIG17-智慧零售测试工程师（CSIG	武汉	技术	2022年1月2日	负责公司级	本科以上学历，5年以上	
8	22989-腾讯云Serverless高级前端开	深圳	技术	2022年1月2日	1、负责腾讯	1、本科及以上学历，计	
9	22989-腾讯云编程语言虚拟机/编译	北京	技术	2022年1月2日	针对腾讯云	本科以上学历，硕士及	
10	CSIG15-智能平台产品部-高级后台开	深圳	技术	2022年1月2日	1、负责翻译	本科以上学历，计算机	

图 3.35　网页中 10 个岗位的基本信息

3. 多网页数据爬取

如果要爬取该职业类所有网页的数据，如图 3.36 所示，该如何操作呢？

图 3.36　多网页数据爬取目标显示

前面讲过，网页翻页功能有时体现在 URL 中，有时会体现在响应资源的参数上继续来分析本网页的参数，如图 3.37 所示，查看下翻页时参数的变化。

图 3.37　查看参数项

在图 3.37 中，查询参数栏中，大部分参数是空的，直接赋空值就可以了。图 3.37 上标记出来的①② 参数分析如下：

① “pageIndex：1”表示的是网页页码，1 表示当前在网页的第 1 页。通过测试可知，翻到第 2 页时，pageIndex 的值为 2。

② “pageSize：10”表示一个网页中显示的岗位数量，这里表示一个网页中默认显示 10 个岗位。

通过分析可知，如果要爬取多个网页数据，只需要修改 pageIndex 的值。可以使用循环遍历，pageIndex 的值从 1～385 页，循环计数器的值作为 pageIndex 值，如图 3.36 所示。

局部修改代码如下：

```
import requests
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "https://careers.tencent.com/tencentcareer/api/post/Query"
for i in range(385): #使用循环控制翻页
    keys = {
        'timestamp': '1638023920104',
        'countryId': '',
        'cityId': '',
        'bgIds': '',
        'productId': '',
        'categoryId': '',
        'parentCategoryId': '40001',
        'attrId': '',
        'keyword': '',
        'pageIndex': i, #变量控制页码
        'pageSize': '10',
        'language': 'zh - cn',
        'area': 'cn'
    }
    response1 = requests.get(url = url, headers = headers, params = keys)
    postinfor = response1.json()

    for i in postinfor['Data']['Posts']:
            print(i['PostId'])
```

程序执行结果，获得全部的招聘岗位 PostId，有了 PostId 就可以爬取每个招聘岗位的详情数据了，这里就不再赘述，由读者自行完成。

下面要爬取不同职业类的所有网页所有岗位的详情数据如图 3.38 所示。

【解析】 要实现多类岗位的爬取，首先确定类别是如何体现的，查看方法和前面相同，这里不再赘述。打开当前网页的 Network 面板，查看每个职类数据的特点，如图 3.39 所示。

打开某一个职类岗位的参数，如图 3.37 所示，分析这两个网页的关联之处。图 3.37 是打开技术类的岗位网页，它的众多参数中有一个 parentCategoryId：40001，通过这个参数控制程序，就可以实现选择不同职业类。也就是说只需要在上一步的基础上，把

图 3.38　多类岗位爬取目标

图 3.39　查看 Preview 数据

parentCategoryId 参数设置为动态参数，这个参数可以是 4001、4002、4003、4005 等，不同值代表不同职业类，只要能通过程序获得职业类的 parentCategoryId，就能控制爬虫程序爬取对应职业类下的所有页所有招聘岗位的详情数据。

那么在这一步中需要进行的操作如下：首先，爬取本页中所有职类的 CategoryID，把 CategoryID 传递给上一步；其次，修改上一步的代码，把 parentCategoryId 变成可变参数；最后，parentCategoryId 值通过 CategoryID 传递过来。

爬取 CategoryID 值和前面其他数据爬取方法相同，它们都是 JSON 数据，爬取 JSON 并解析并提取出 CategoryID，代码如下：

```
import requests
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "https://careers.tencent.com/tencentcareer/api/post/ByHomeCategories"
params = {
    'timestamp': '1641133718615',
    'num': '6',
```

```
    'language': 'zh-cn'
}
response0=requests.get(url = url,headers = headers,params = params)
Cateinfor =response0.json()
for categ in Cateinfor['Data']:
    print(categ['CategoryId'])
```

程序执行结果如下：

```
40001
40003
40006
40002
40005
40008
```

有了 CategoryID，把它放在列表中，通过循环实现职业类别的遍历，这里不再赘述，由读者自行完成。

【例 3-9】 爬取网站 http://scxk.nmpa.gov.cn：81/xk/上所有企业的化妆品生产许可证详情数据，爬取目标如图 3.40 所示。

图 3.40 待爬取的目标数据

主要解题思路分析如下：

第①步：网页中每个企业名称对应一个超级链接。

第②步：单击企业名称进入各个对应企业的化妆品许可证详情网页。

第③步：在许可证详情网页定位目标数据的响应资源 Headers 面板，获取各类参数，编写爬虫程序，爬取此网页的目标数据。

第④步：建立多企业标题链接与化妆品许可证详情网页的对应关系，实现爬取这一网页所有企业的许可证详情数据。

第⑤步：翻页功能实现爬取所有企业的化妆品许可证详情数据。

代码可以从内到外来写，先实现单个企业的许可证信息爬取，再爬取企业标题页上所有企业的许可证信息，最后实现翻页去爬取所有页所有企业对应的许可证详情数据，下面分步骤实现上述过程。

1. 单个企业化妆品许可证详情数据爬取

爬取目标是爬取单个企业的化妆品生产许可证详情数据，目标数据的定位如 3.3 节所述，这里不再赘述。

定位到当前网页的目标数据，如图 3.41 所示。在 Preview 面板预览可知，目标数据是动态加载的 JSON 数据。通过查看响应资源 Headers 面板可知，此网页是带参数的 POST 请求。

图 3.41　定位目标数据的 Headers

代码如下：

```
import requests
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "http://scxk.nmpa.gov.cn:81/xk/itownet/portalAction.do?method = getXkzsById"
keys = {
    'id': 'af4832c505b749dea76e22a193f873c6'
}
response1 = requests.post(url = url,headers = headers,data = keys)
response1.json()
```

程序运行结果如图 3.42 所示。

2. 单个网页多链接数据爬取

目标数据是爬取当前企业标题网页上所有企业的化妆品许可证详情数据，爬取目标如图 3.43 所示。

企业浏览网页与其化妆品许可证详情网页是什么关系呢？与 3.3 节爬取多个招聘岗位链接进入对应的具体招聘信息一样，这个对应链接关系是通过 Headers 面板下的参数来控制的。

打开每个企业的化妆品许可证详情网页查看后会发现，它们具有相同的 Request URL

{'businessLicenseNumber': '914101057457968673',
'businessPerson': '姜长宏',
'certStr': '一般液态单元（护发清洁类）；气雾剂及有机溶剂单元（有机溶剂类）',
'cityCode': '',
'countyCode': '',
'creatUser': '',
'createTime': '',
'endTime': '',
'epsAddress': '郑州市金水区沙口路113号',
'epsName': '郑州付氏育发化妆品有限公司',
'epsProductAddress': '郑州市金水区沙口路113号',
'id': '',
'isimport': 'N',
'legalPerson': '姜长宏',
'offDate': '',
'offReason': '',
'parentid': '',
'preid': '',
'processid': '20211022080707202rw1bp',
'productSn': '豫妆20160033',

图 3.42　单个企业的化妆品许可证详情数据中 JSON 数据部分截图

企业名称	许可证编号	发证机关	有效期至	发证日期
深圳福雅化妆品有限公司	粤妆20210419	广东省药品监督管理局	2026-12-30	2021-12-31
深圳市宝莱化妆品有限公司	粤妆20210418	广东省药品监督管理局	2026-12-30	2021-12-31
宝丽（广东）生物科技有限公司	粤妆20200020	广东省药品监督管理局	2025-01-14	2021-12-31
广东施露兰化妆品有限公司	粤妆20160331	广东省药品监督管理局	2026-12-30	2021-12-31
平舆玛雅生物工程有限公司	豫妆20170001	河南省药品监督管理局	2026-12-30	2021-12-31
洛阳科迪艺思化妆品有限公司	豫妆20160061	河南省药品监督管理局	2026-12-30	2021-12-31
平舆冰王生物工程有限公司	豫妆20160038	河南省药品监督管理局	2026-12-30	2021-12-31
郑州付氏育发化妆品有限公司	豫妆20160033	郑州市市场监督管理局	2026-12-31	2021-12-31
南阳市广秀保健品有限责任公司	豫妆20160027	河南省药品监督管理局	2026-12-30	2021-12-31
镇平人仁实业有限公司	豫妆20160024	河南省药品监督管理局	2026-12-30	2021-12-31
西施兰(南阳)药业股份有限公司	豫妆20160016	河南省药品监督管理局	2026-12-30	2021-12-31
河南汉方药业有限责任公司	豫妆20160009	郑州市市场监督管理局	2026-12-31	2021-12-31
河南所爱化妆品有限公司	豫妆20160003	河南省药品监督管理局	2026-12-30	2021-12-31
新疆金海娜生物科技有限公司	新妆20160016	新疆维吾尔自治区药品监督管理局	2026-12-30	2021-12-31
江苏金泽生物科技有限公司	苏妆20170002	江苏省药品监督管理局	2026-12-30	2021-12-31

第1/380页，15条/页，总共【5694】条数据　首页 上一页 2 3 4 5 6 7 下一页 尾页

图 3.43　爬取目标数据

和 Request Method，不同的请求参数“id：c1acc81f9d88478cabc0ddcd9a11ee2d”，如图 3.44 和图 3.45 所示。在写爬虫程序时，只需要传递不同的请求参数就可以共享同一个爬虫程序，参数不同，爬取的数据对应不同的网页数据。显然，这个 ID 对应的就是每个链接企业化妆品许可证详情网页的 ID，只要能找到企业的 ID 就能获得其对应的许可证详情网页信息。下面首先去爬取每个企业的 ID，然后把 ID 作为参数传递到上一步获取单个公司的化妆品许可证详情数据的爬虫程序中。

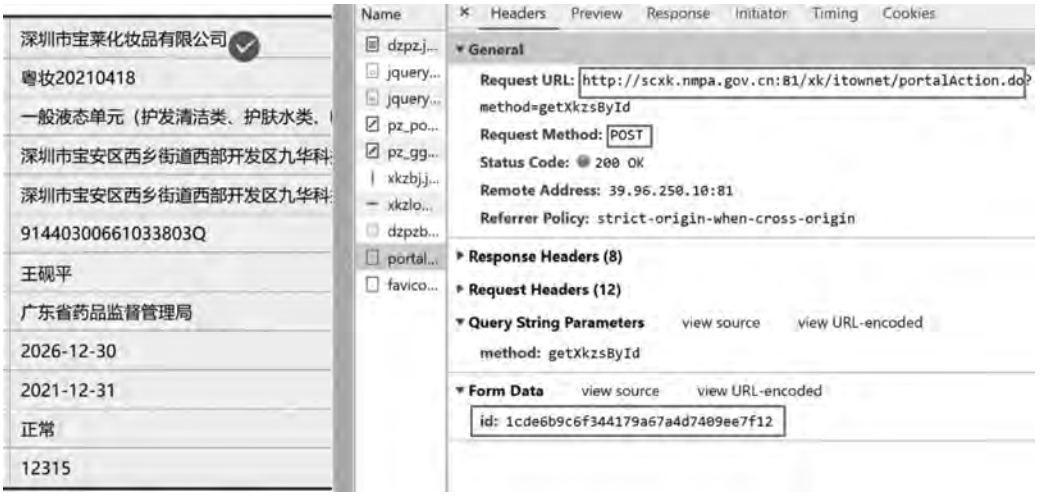

图 3.44　企业 1 的化妆品许可证详情数据 Headers 查看

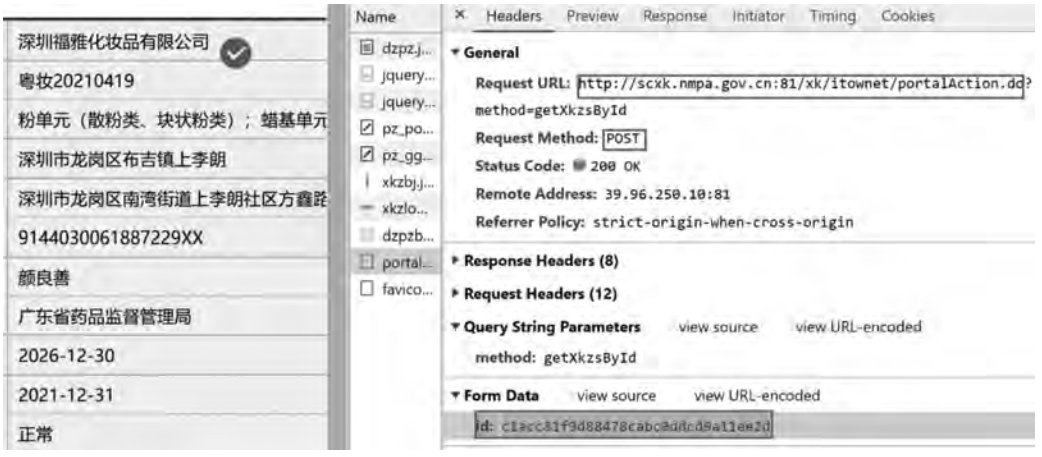

图 3.45　企业 2 的化妆品许可证详情数据 Headers 查看

打开并分析浏览器企业的网页，查找企业的 ID 数据。操作步骤如下，对应如图 3.46 所示：

第①步：打开 Network 面板。

第②步：打开控制器的搜索工具。

第③步：在弹出的搜索框里，输入爬取目标中的任意几个字。

第④步：单击搜索到的多层级资源中最里层资源。

第⑤步：查看 Preview 面板中是否有需要爬取的数据，如果有，那就对此资源发起请求。

第⑥步：查看资源的 Headers 信息，为爬虫程序作准备，如图 3.47 所示。

第⑦步：解析爬取到的数据，解析并提取出其中的企业 ID。

代码如下：

```
import requests
```

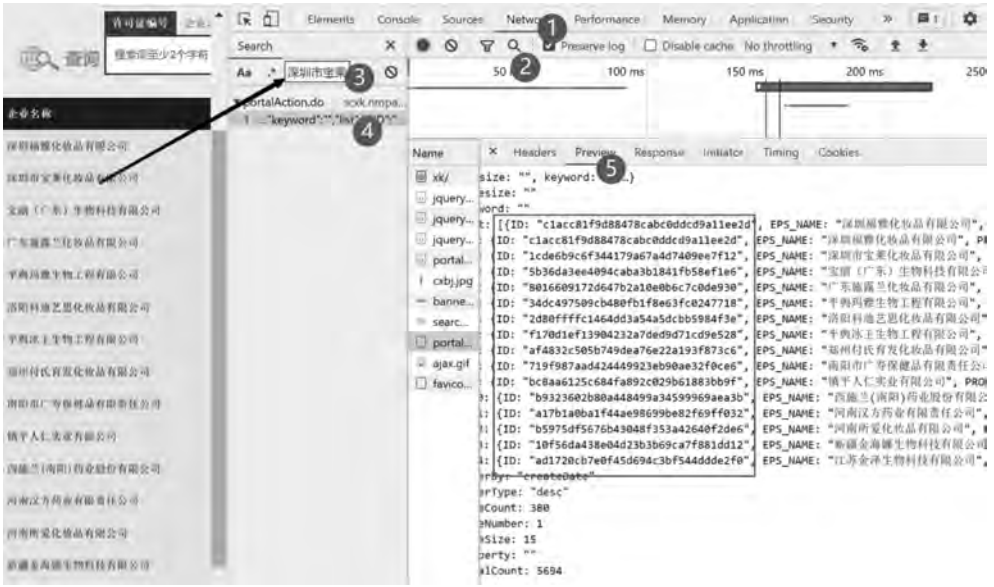

图 3.46　定位并查看目标数据的 Preview 预览信息

图 3.47　查看目标数据的 Headers 头部信息

```
import json
url = 'http://scxk.nmpa.gov.cn:81/xk/itownet/portalAction.do?method=getXkzsList'
headers = { 'User - Agent ': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36'}
data = {'on': 'true',
    'page':1,
```

```
    'pageSize':'15',
    'productName':'',
    'conditionType':'1',
    'applyname': ''
     }
response = requests.post(url = url,data = data,headers = headers)
js_id = response.json()
for idinfor in js_id[ 'list']:#解析 JSON
    print(idinfor['ID'])
```

程序执行结果为一个网页中所有企业的 ID 数据,代码如下:

```
c1acc81f9d88478cabc0ddcd9a11ee2d
1cde6b9c6f344179a67a4d7409ee7f12
5b36da3ee4094caba3b1841fb58ef1e6
8016609172d647b2a10e0b6c7c0de930
34dc497509cb480fb1f8e63fc0247718
2d80ffffc1464dd3a54a5dcbb5984f3e
f170d1ef13904232a7ded9d71cd9e528
af4832c505b749dea76e22a193f873c6
719f987aad424449923eb90ae32f0ce6
bc8aa6125c684fa892c029b61883bb9f
b9323602b80a448499a34599969aea3b
a17b1a0ba1f44ae98699be82f69ff032
b5975df5676b43048f353a42640f2de6
10f56da438e04d23b3b69ca7f881dd12
ad1720cb7e0f45d694c3bf544ddde2f0
```

有了企业 ID,遍历循环调用上一步中爬取单个企业的化妆品许可证详情数据的代码。修改上一步的代码如下:

```
import requests
headers = {
    'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "http://scxk.nmpa.gov.cn:81/xk/itownet/portalAction.do?method = getXkzsById"
def singecomp(compid):#函数实现爬取单个指定 ID 企业的化妆品许可证详情数据
    keys = {
        'id': compid #参数为可变参数
    }
    response1 = requests.post(url = url,headers = headers,data = keys)
    return response1.json()
```

把爬取的企业 ID 放在列表中,循环调用函数 singecomp(compid),把 ID 的值传递给 compid,就爬取到了不同企业的化妆品许可证详情数据。

3. 多个网页数据爬取

网页连续翻页可能会体现在 URL 中,也可能会体现在参数中。如图 3.48 所示,查看企业信息标题网页,找到资源数据对应的 Headers 信息,查看参数中是否有控制网页翻页功能的参数,在 Headers 信息有两个重要参数如下:

“page: 1”表示当前在第 1 页,当翻到下个网页时,page 值为 2,因此 page 参数用来实现

网页翻页操作。

“pageSize：15”表示在一页网页上默认显示有 15 家企业。

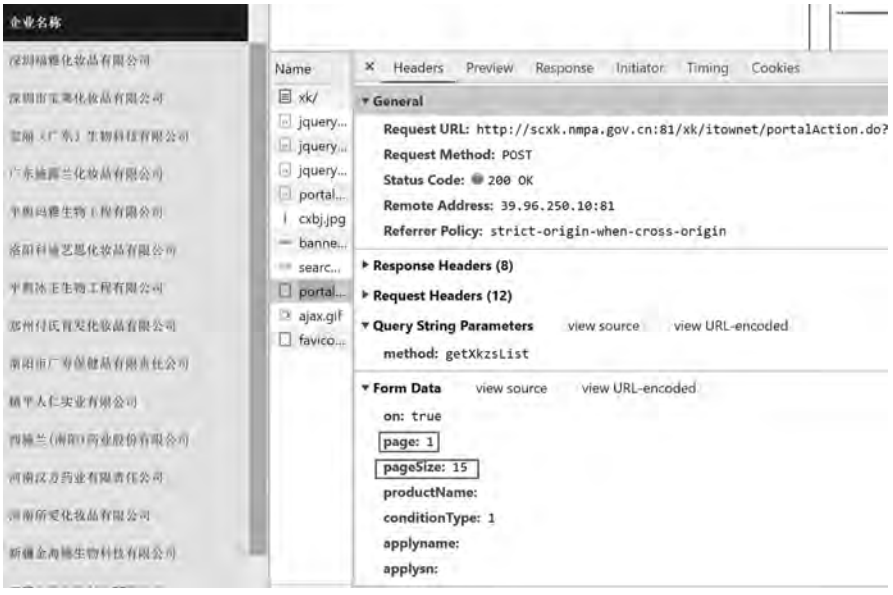

图 3.48　查看网页翻页参数

通过分析可知，参数 page 控制要爬取数据所在的页码，如果值为 1，那么爬取的就是第 1 个网页的数据，如果值为 2，爬取的是第 2 个网页的数据。设置一个循环遍历，遍历 page 值，实现对所有页的数据爬取。

单个网页可以使用上述调用函数的形式，也可以不用调用函数形式，直接对 URL 发送请求，代码如下：

```
import requests
import json
url = 'http://scxk.nmpa.gov.cn:81/xk/itownet/portalAction.do?method = getXkzsList'
headers = { 'User - Agent ': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/85.0.4183.121 Safari/537.36'}
for i in range(15):
    page = i
    data = {'on': 'true',
    'page':page,                          #页，循环控制所有网页
    'pageSize':'15',                      #每个网页的条目数
    'productName':'',
    'conditionType':'1',
    'applyname': ''
    }
    response = requests.post(url = url,data = data,headers = headers)
    js_id = response.json()               #字典类型
    id_list = []                          #存放企业的 ID
    for dic in js_id['list']:
        id_list.append(dic['ID'])         #批量爬取了每个企业的 ID
    fp = open('.\许可证.json','w',encoding = 'utf - 8')
```

```
url1 = 'http://scxk.nmpa.gov.cn:81/xk/itownet/portalAction.do?method = getXkzsById'
for id in id_list:#循环爬取所有企业的化妆品许可证详情数据
    data1 = {'id':id}
    response1 = requests.post(url = url1,data = data1,headers = headers)
    print(response1.json())
    json.dump(response1.json(),fp,ensure_ascii = False)          #保存数据
```

程序运行结果，得到一个包含网站中所有公司的许可证信息的 JSON 文件，该 JSON 文件内容部分截图如图 3.49 所示。

图 3.49　爬取得到的所有数据的 JSON 数据

3.4　POST 请求的两种参数格式

POST 请求通常有 Form Data 和 Request Payload 两种参数类型。

3.4.1　Form Data 类型

图 3.50 是一个 Form Data 格式的参数类型，这类参数在写爬虫程序的时候有两种处理方式。

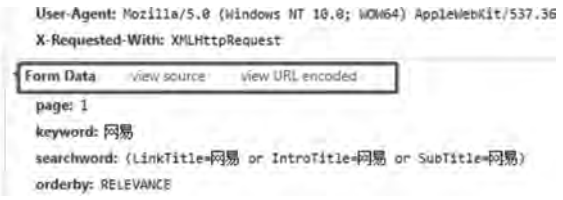

图 3.50　Form Data 格式的参数类型

第 1 种是把这个 POST 请求变成 GET 请求，即把请求参数通过"?key1＝value1 &key2＝value2"拼接在 URL 当中，然后以 GET 方式请求就可以了，请求方式如下：

```
response = requests.get ( url, headers = headers)
```

其中，URL 为拼接的 URL。

第 2 种是仍然发送 POST 请求，将参数放在 data 参数中，请求方式如下：

```
response = requests.post(url, headers = headers, data = data)
```

其中，URL 中不携带参数。

这两种方法，建议使用第 2 种，因为这种方法参数作为变量时，设置更灵活。

3.4 节中的案例就是 data 格式的参数，如图 3.51 所示，爬虫代码实现如 3.4 节所述。

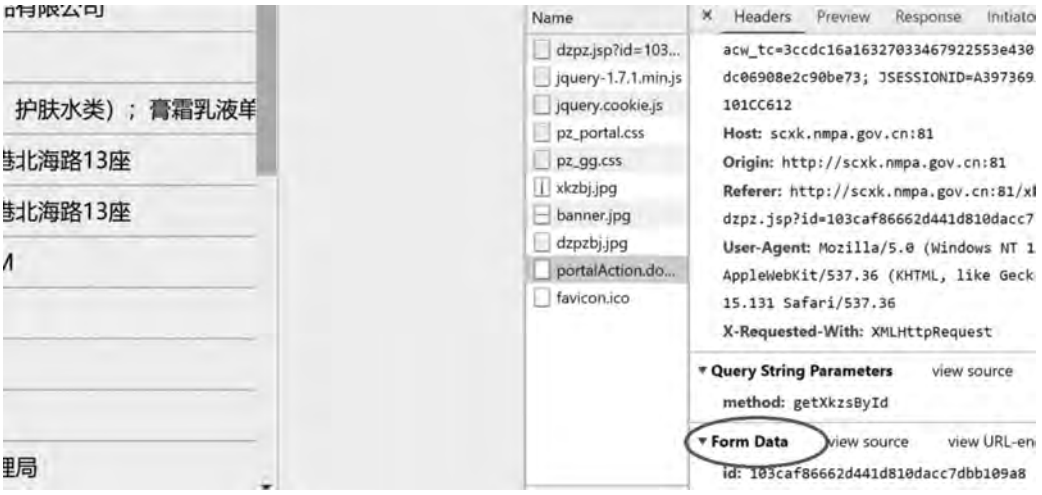

图 3.51　POST 请求的 Form Data 参数格式

3.4.2　Request Payload 类型

Request Payload 参数为自动变成了 JSON 类型，此时必须发 POST 请求，将 JSON 对象传入才可爬取数据，如图 3.52 所示。

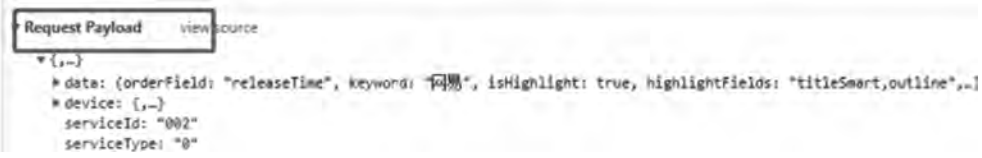

图 3.52　Request Payload 参数类型

这种参数的请求方式如下：

```
response = requests.post(url,json = data,headers = headers)
```

其中参数 data 一定要序列化。

【例 3-10】 从国家电网电子商务平台爬取某个公告基本信息，爬取目标如图 3.53 所示。

操作步骤和 3.3 节相同，这里不再赘述，唯一不同的是请求参数类型为 Request Payload，代码如下：

图 3.53　Request Payload 参数格式

```
import requests
headers = {
    'User - Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.102 Safari/537.36' }
url = "https://ecp.sgcc.com.cn/ecp2.0/ecpwcmcore//index/getNoticeBid"
key = "2021121535839657"
response = requests.post(url = url,json = key,headers = headers)
response.json()
```

程序运行的结果如图 3.54 所示。


```
{'successful': True,
 'resultValue': {'fileFlag': '1',
  'notice': {'PURPRJ_NOTICE_DET_ID': None,
   'CONTACT': '雷工',
   'PURPRJ_STATUS': 130020016,
   'PURPRJ_NAME': '国网湖南省电力有限公司35千伏及以上输变电工程设计施工监理2021年度第二次资格预审',
   'TAX': '0731-85337512',
   'PRJ_INTRODUCE': None,
   'PRJ_STATUS': 0,
   'PURPRJ_ID': 2021121534526676,
   'CHG_NOTICE_CONT': ' ',
   'PUR_TYPE': 130007002,
   'ORG_TYPE': 4,
   'PUBLISH_ORG_NAME': '国网湖南省电力有限公司',
   'IS_SELF_EXEC': 100038001,
   'PAY_MODE_NAME': '线下支付',
   'OPENBID_ADDR': '不开标',
   'BID_AGT': '湖南湘能创业项目管理有限公司',
   'IMPL_MODE': 130022002,
   'PUR_TYPE_NAME': '服务',
   'NOTICE_TYPE': 100063003,
   'ONLINE_BID_NOTICE_ID': 2021121535839657,
   'BID_AGT_ADDR': '湖南省长沙市天心区新韶东路379号康园会所三楼（西侧）',
   'PAY_MODE': 130045002,
```


图 3.54　程序爬取到的 JSON 数据部分截图