

技术报告

目录

技术报告

网络爬虫

1. 应用背景

- 1.1 网络爬虫基本知识
- 1.2 HTML语言
- 1.3 路径
- 1.4 文件后缀名
- 1.5 应用编程接口
- 1.6 可拓展标记语言
- 1.7 编码
- 1.8 反爬虫
- 1.9 JavaScript
- 1.10 JSON
- 1.11 URL编码
- 1.12 Cookie

2. 技术细节

- 2.1 requests模块
- 2.2 解析模块Beautiful Soup
- 2.3 User-Agent
- 2.4 按属性查询
- 2.5 文件读写
- 2.6 爬取图片并保存
- 2.7 编码属性获取
- 2.8 查找JS文件
- 2.9 节点选择器
- 2.10 获取子结点

3. 项目应用示例 —— 爬取网易云音乐

网络爬虫

1. 应用背景

1.1 网络爬虫基本知识

网络爬虫（Web Crawler），网络爬虫是按照一定规则自动抓取网页信息的程序。如果把互联网比作一张大网，把蜘蛛网的节点比作一个个网页。那么爬虫就是在网页上爬行的蜘蛛，每爬到一个节点就能够访问该网页的信息，所以又称为网络蜘蛛（Web Spider）。日常浏览的网页中，既有图片、文字，还有精致的排版，这些页面都依靠源代码^[1]的功劳，源代码会定义每个标题、段落、图片等排版，浏览器通过解析源代码，呈现出网页画面。所以，爬虫获取的就是浏览器解析之前的源代码，也就是图-1的内容。

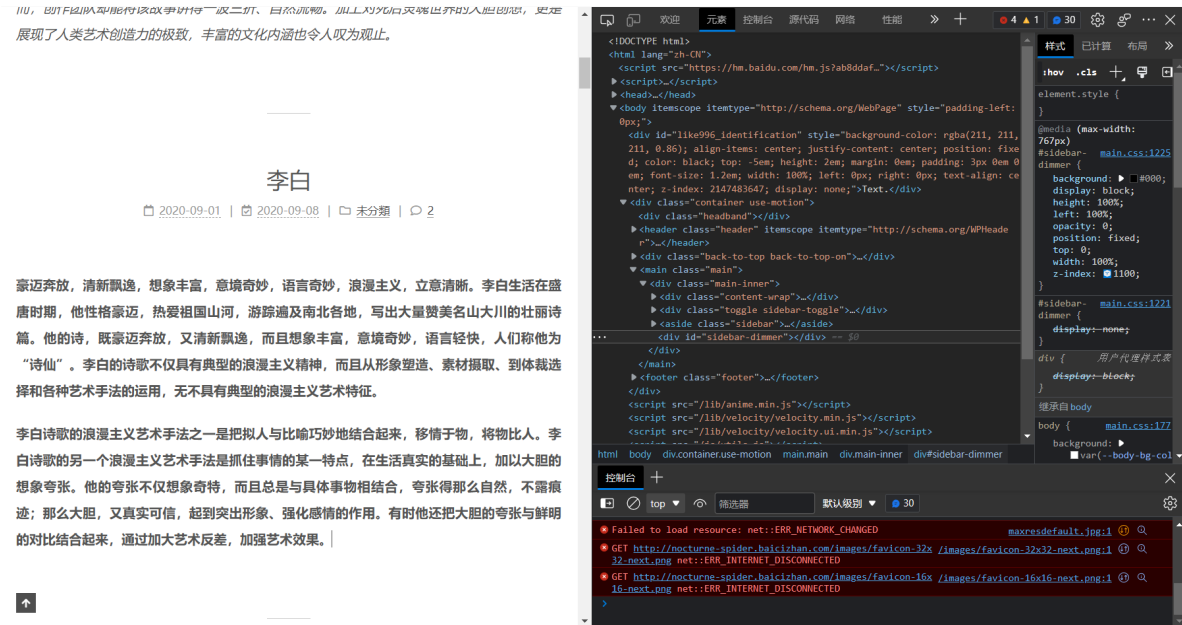


图-1 解析前的源代码（右边）

例如，若要通过某宝评论信息分析出《榴莲味口香糖》值不值得买，不使用爬虫技术，就需要先打开网页，然后找到评论信息，再一条一条的翻看。而对于网络爬虫来说，它能够自动化获取《榴莲味口香糖》网页的所有信息，通过提取网页中的评论内容，将信息保存到文档中，便于对数据进行查看和分析。所以，网络爬虫就是自动化从网页上获取信息、提取信息和保存信息的过程。

日常我们访问每个网站都是通过链接打开的。这里的链接也叫做URL，URL全称为Uniform Resource Locator，即**统一资源定位符**，指定了我们要查找资源的地址。一般来说，URL的信息包含**主机名**和**访问协议**。主机名（英文：hostname）就是要访问的计算机的名字，两个URL中主机名不同，访问的网页也不同。常见的访问协议包括**超文本传输协议**（英文：HyperText Transfer Protocol，简称：http），HTTP协议是互联网数据传输的一种规则，它规定了数据的传输方式，就像是我们寄快递选择的快递公司，快递公司规定了邮寄方式。HTTP协议定义了**客户端**^[2]和**服务器**^[3]之间传递消息的

内容和步骤，就像快递公司，定义了客户和仓库之间发送快递的内容和步骤。当URL的**协议部分**^[4]写的是http时，表明服务器传输数据使用的是HTTP协议。另一种常见的访问协议是**超文本传输安全协议**（英文：HyperText Transfer Protocol Secure，简称：https），HTTP协议在进行数据传输时，内容是未加密的，传输内容可能被窃听或篡改，安全性比较差，而HTTPS并非是全新的协议，只是在传输之前加了一层**保护**^[5]，让内容安全不易被窃听。所以说，HTTPS协议是HTTP的安全版，使用HTTPS传输能够让传输的数据更安全。对于一个URL来说，"//"为分隔符，表示后面的字符串是主机名，主机名后面的"/"表明，要在后面写上文件地址，如果不写一般默认为主页，我们打开一个网页，"/"后面的名称不同，链接到的页面也就不同。最终，URL的组成结构可以总结为：访问协议 + : + // + 主机名 + / + 文件路径名（可省略）。

我们访问的网页资源是存储在**服务器**中的。服务器可用于**管理资源并为用户提供服务**，其特点就是运算速度快，能为大量用户服务。服务器的种类有很多，当浏览网页时其主要作用就是将网页信息提供给浏览器，此时的服务器也被称为**Web服务器**。

对于HTTP协议的请求和响应过程，浏览器会先发送HTTP请求，告诉Web服务器需要的数据，Web服务器收到请求后，按照请求执行，并返回HTTP响应消息，浏览器收到返回的数据后，会将源代码解析成网页展示出来。HTTP发送的**请求**（Request）消息主要包含两部分“对什么”和“怎么做”。“对什么”是我们前面学习的URL，就是要访问的目标。怎么做”一般叫做**方法**（Method），是指让Web服务器完成什么工作。由于浏览器发送请求时，将“对什么”和“做什么”信息放在头部，所以，存放这些信息的地方又叫**请求头**（Request Headers）。Web服务器收到请求消息后，会根据请求进行处理。并将**响应**（Response）消息返回给浏览器。响应消息的头部叫做**响应头**（Response Headers），响应头中的数据用于告诉浏览器此次请求执行失败还是成功。响应头中用于告知浏览器执行结果成功或失败的叫做**状态码**。状态码是由3位的数字构成的，主要用于告知客户端的HTTP请求的执行结果。状态码可以让我们了解到服务器是正常执行结果，还是出现了错误。

表-1 常见的状态码

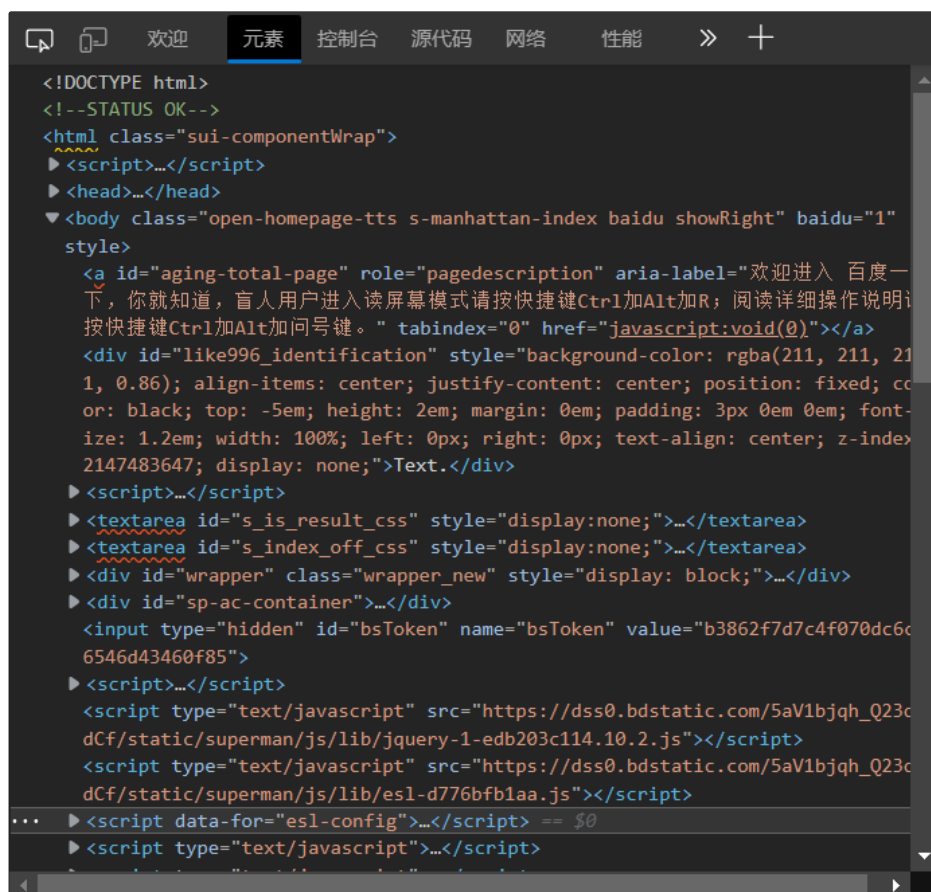
状态码	含义
1XX	告知请求的处理进度和情况
2XX	成功
3XX	表示需要进一步操作
4XX	客户端错误
5XX	服务器错误

当浏览器发送HTTP请求且Web服务器执行了请求后，若返回的响应头中状态码为200，表示执行成功，浏览器此次的请求正常执行。日常访问网页时，也会遇见状态码：404，404 (Not Found) 表示服务器无法找到请求的资源，或者，有的服务器拒绝你的请求并不想说明理由时也会提示404。或者，有时候打开网页时会提示状态码503，状态码503 (Service Unavailable) 表示服务器处于超负荷状态或正在进行停机维护，现在无法处理浏览器的请求。

爬虫能够帮助我们自动化的获取网页信息，但是，网络资源也会带来很多问题。一是影响服务器性能，爬虫主要请求服务器的资源，大量快速的访问服务器，会影响服务器速度，耗费服务器性能。二是法律风险，图片、视频或摄影作品等大部分是有版权的，将抓取的内容商业化也可能带来风险。网络资源虽然非常丰富，但我们在使用爬虫获取网络资源时，需要遵循网络的基本规则——**robots协议**。这个协议一方面是一个爬虫技术人员需要遵守的道德准则，另一方面，如果将爬取结果商用并获取利益，还会面临法律风险。

1.2 HTML语言

超文本标记语言（英文：HyperText Markup Language，简称：HTML语言），它用来定义网页内容和结构。HTML是由一系列的**标签**组成，这些标签组合起来就是我们浏览器看到的网页。



```
<!DOCTYPE html>
<!--STATUS OK-->
<html class="sui-componentWrap">
  <script>...</script>
  <head>...</head>
  <body class="open-homepage-tts s-manchattan-index baidu showRight" baidu="1" style>
    <a id="aging-total-page" role="pagedescription" aria-label="欢迎进入 百度一下，你就知道，盲人用户进入读屏幕模式请按快捷键Ctrl加Alt加R；阅读详细操作说明请按快捷键Ctrl加Alt加问号键。" tabindex="0" href="javascript:void(0)"></a>
    <div id="like996_identification" style="background-color: rgba(211, 211, 211, 0.86); align-items: center; justify-content: center; position: fixed; color: black; top: -5em; height: 2em; margin: 0em; padding: 3px 0em 0em; font-size: 1.2em; width: 100%; left: 0px; right: 0px; text-align: center; z-index: 2147483647; display: none;">Text.</div>
    <script>...</script>
    <textarea id="s_is_result_css" style="display:none;">...</textarea>
    <textarea id="s_index_off_css" style="display:none;">...</textarea>
    <div id="wrapper" class="wrapper_new" style="display: block;">...</div>
    <div id="sp-ac-container">...</div>
    <input type="hidden" id="bsToken" name="bsToken" value="b3862f7d7c4f070dc6c6546d43460f85">
    <script>...</script>
    <script type="text/javascript" src="https://dss0.bdstatic.com/5aV1bjqh_Q23c...dCf/static/superman/js/lib/jquery-1-edb203c114.10.2.js"></script>
    <script type="text/javascript" src="https://dss0.bdstatic.com/5aV1bjqh_Q23c...dCf/static/superman/js/lib/esl-d776bfb1aa.js"></script>
    ...
    <script data-for="esl-config">...</script> == $0
    <script type="text/javascript">...</script>
```

图-2 HTML语言

标签是用来标记内容块的，主要有两种形式：成对出现和单独出现。成对出现的标签形式如：<开始标签>内容<结束标签>，结束标签只比开始标签多一个斜杠"/"。例如，<html></html>表明这是一个网页文档、<head></head>标签用于定义文档的头部信息，这些信息不会展示在网页中、<body></body>标签用于定义文档的主体，包含网页的图片、文字、链接、视频等多种展现形式。单独标签，如
表示换行。还有值得注意的地方是，标签是可以嵌套的。

属性是用来丰富表现形式的，一般放在开始标签里，并且以属性名="属性值"的形式展现，属性还可以描述内容的颜色、边框等等。如图-2中的便包含了5个属性。

表-2 常见的属性及其解释

属性	解释	值
align	对齐方式	center/left/right
bgcolor	背景颜色	yellow/green
border	表格边框	1/2/3
class	规定元素的类名	classname
id	规定元素的唯一id	id

想要获取网页中的数据，首先要获取网页 HTML 代码，再把数据从中提取出来。我们要向网页的服务器发送请求，服务器返回的响应就是网页 HTML 代码。

1.3 路径

路径（Path）用来表示文件或文件夹的位置。当需要访问文件或文件夹时，路径就像现实中的地址一样，帮助找到目标文件或文件夹在什么位置。常用的计算机系统分为Windows系统 和 macOS系统。在Windows系统的路径中，使用反斜线（ \ ）分隔各个文件夹和文件名。同时，在路径的最前面，是盘符的字母和一个英文冒号，表示文件或文件夹具体是在哪个盘的路径下。

而在macOS系统中，文件夹和文件名使用正斜线（ / ）进行分隔。需要注意的是，macOS系统中，没有盘符的概念，所有的路径都是从根目录（ / ）开始。



图-3 系统路径示例

1.4 文件后缀名

文件后缀名又叫文件扩展名 (File extension) , 可以用来判断某个文件属于什么类型。在文件名中, 点号, 也就是英文句号 (.) 和其后面的部分即为文件后缀名。文件后缀名不区分大小写。



图-4 文件后缀名

一些常见的文件后缀名如表-3所示。

表-3 常见文件后缀名

文件类型	后缀名
图片文件	.jpg .jpeg .gif .png .bmp
文本文件	.txt
PDF文件	.pdf
Word文件	.docx .doc
Excel文件	.xlsx .xls
PPT文件	.pptx .ppt

1.5 应用编程接口

应用编程接口（英文：Application Programming Interface，缩写：API），通俗易懂的说，API 其实就是别人已经写好的可以实现特定功能的函数，只需要调用它，传入规定的参数，这个函数就可以实现功能。

可以把 API 的诞生用一个小故事来展示：程序员A开发了软件A，程序员B正在研发软件B。有一天，程序员B想要调用软件A的部分功能，但是他又不想从头看一遍软件A的代码，怎么办呢？程序员A想了一个好主意：我把软件A里你需要的功能打包好，写成函数；你把这个函数放在软件B里，就能直接用我的功能啦。其中，API 就是程序员A说的那个函数。

其实日常生活中，有很多 API 的场景。比如说，当我们在某宝上下单后，店家选用顺丰发货，明明某宝和顺丰是两家独立的公司，但为什么在某宝上能查看到顺丰的物流信息？这就是API的功劳。当查看快递信息时，某宝利用顺丰提供的 API，可以实时调取信息呈现在自己的网站上。很多网站都会提供不同功能的 API，如果网站提供了，我们就不需要去爬取 HTML 代码，可以直接去网站的 API 获取数据，这样做既减轻了网站的服务器压力，也降低了我们解析网页的难度。通常，API 一般以 URL 形式存在，我们可以通过访问这些 URL 来获取数据。

1.6 可扩展标记语言

可扩展标记语言（英文：Extensible Markup Language，简称：XML）是一种标记语言。标记指计算机所能理解的信息符号，有了标记，我们可以通过程序很方便的识别数据的类型和层次结构。比如图中的 Tom 和 Jerry 是数据内容，<to> 和 <from> 用来标记 Tom 和 Jerry，表示他们分别是收件人和发件人。

代码

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <note>
3   <to>Tom</to>
4   <from>Jerry</from>
5   <heading>Remember</heading>
6   <body>Catch Me If You Can</body>
7 </note>
```

图-5 标记语言

HTML 使用标记来注明文本、图片和其他内容，以便于在 Web 浏览器中显示，HTML 标记包含固定的标签如 <head>，<title>，<body>，<p>，<div>，， 等等。而在 XML 中可以定义自己的标签和文档结构，没有固定的标签。

<div>HTML:predifined tags</div> <div>HTML1</div> <div><div><h1>title</h1></div><div><p>paragraph</p></div><div><p>paragraph</p></div></div> <div>HTML2</div> <div><div><h1>title</h1></div><div><p>paragraph</p></div><div><p>paragraph</p></div></div>	<div>XML:self-difined tags</div> <div>XML1</div> <div><div><headline>title </headline></div><div><paragraph> paragraph</paragraph></div><div><paragraph> paragraph</paragraph></div></div> <div>XML2</div> <div><div><headline>title </headline></div><div><paragraph> paragraph</paragraph></div><div><paragraph> paragraph</paragraph></div></div>
---	--

图-6 HTML语言和XML语言

XML 被设计用来传输和存储数据，其焦点是**数据的内容**。HTML 被设计用来显示数据，其焦点是**数据的外观**。

1.7 编码

计算机只能处理二进制数字，也就是0或者1，而人能看懂的是一些有意义的字符。因此，需要将人类能看懂的语言字符转换为计算机能看懂的数字序列（0 1序列），这个过程就是**编码**。世界上有很多种语言，每种语言都有自己的特点，为了让计算机看得懂人类语言，就出现了很多种编码方式，常见的有：ASCII，ISO-8859-1，GBK，Unicode，UTF-8 等。

表-4 常用的字符编码集

编码	中文名	字符集数量	特点
ASCII	美国信息交换标准代码	128	仅适用于现代英语
ISO-8859-1		256	是ASCII的拓展
GB2312	信息交换用汉字编码字符集	6763	应用非常广泛
GBK	汉字内码拓展规范	21886	完全兼容GB2312
GB18030	国家标准GB 18030-2005 《信息技术 中文编码字符集》	70244	国标，编码空间庞大
UTF-16			所有字符都是2个字节，不兼容ASCII
UTF-8			变长编码，可节约空间，兼容ASCII

1.8 反爬虫

随着网络的发展和网络爬虫技术的普及，为了收集某些需要的信息，大家会使用网络爬虫进行数据抓取。但是一些爬虫程序会在短时间内发出大量请求，增加网站服务器的过重负担；甚至利用爬虫去盗窃核心数据，导致企业失去竞争力。企业为了保护自身，就在网站里设置了反爬虫机制。常见的反爬虫机制包括：1) 基于 User-Agent 字段的反爬虫。User-Agent 是 HTTP 请求头中用来识别用户身份的一个字段。每个正常用户使用浏览器正常访问网站时都会有标识自身属性的 User-Agent。而服务器如果发现 User-Agent 为空或不正常，则一般会拒绝返回数据，并返回错误码 403 表示禁止访问。2) 基于用户行为的反爬虫。短时间内使用**同一IP地址**^[6]非常频繁的访问同一网站的不同页面，网站服务器发现后会对这个IP地址进行封禁。简单来说，反爬虫是一种用于服务器来识别访问请求是否是正常用户请求的机制。当服务器发现非正常用户的请求时，拒绝其访问服务器资源。

虽然网站有反爬虫机制，我们还是可以在合理范围内，运用些小方法绕过反爬虫，获取到想要的信息。比如说，在程序中设置请求头中的 User-Agent，将爬虫程序伪装成浏览器的正常访问。还可以设置每次访问的间隔时间，来规避短时间内频繁访问的问题。即抓取网页之后，停顿一段时间，再进行下一次抓取。

还有一种反爬虫方式：字体反爬虫。所谓字体反爬虫，就是一些关键数据在网页上查看时，它是正常显示的；而当查看网页的 HTML 代码时，却呈现出一个个的方块。这样就没办法将网页上的数据直接爬取下来。虽然有字体反爬虫，也有办法破解它。HTML 代码中的小方格与网页中数字的对应一定存在某种规则。这种规则是网站自己的一套编码方式，因此这里可以绕道而行——通过一种特殊的字体，将小方格显示成字符，找到小方格原数字与字符之间的关系后，统一替换成正常的数字。一种具体的方法是先把小方格用其他的编码方式重新编码，比如说“UTF-8”编码，找到编码后的二进制数据与网页数字之间的对应关系，然后将它们进行替换。接着将二进制数据，再转换成字符串，就可以破解字体反爬虫机制，获取到数据。

1.9 JavaScript

JavaScript缩写JS，是一门编程语言，当应用在HTML文档时，可为网站提供动态交互效果，常在网页上见到的游戏、动画、按钮、表单数据等都会应用到JS语言。如果与人体结构类比的话，HTML就是骨骼，用于描述网页结构，JavaScript就是肌肉，可以对外界刺激做出反应。HTML是一种标记语言，而JS是一种动态编程语言。HTML在网页中提供静态内容，JS负责向静态网页添加动态功能。

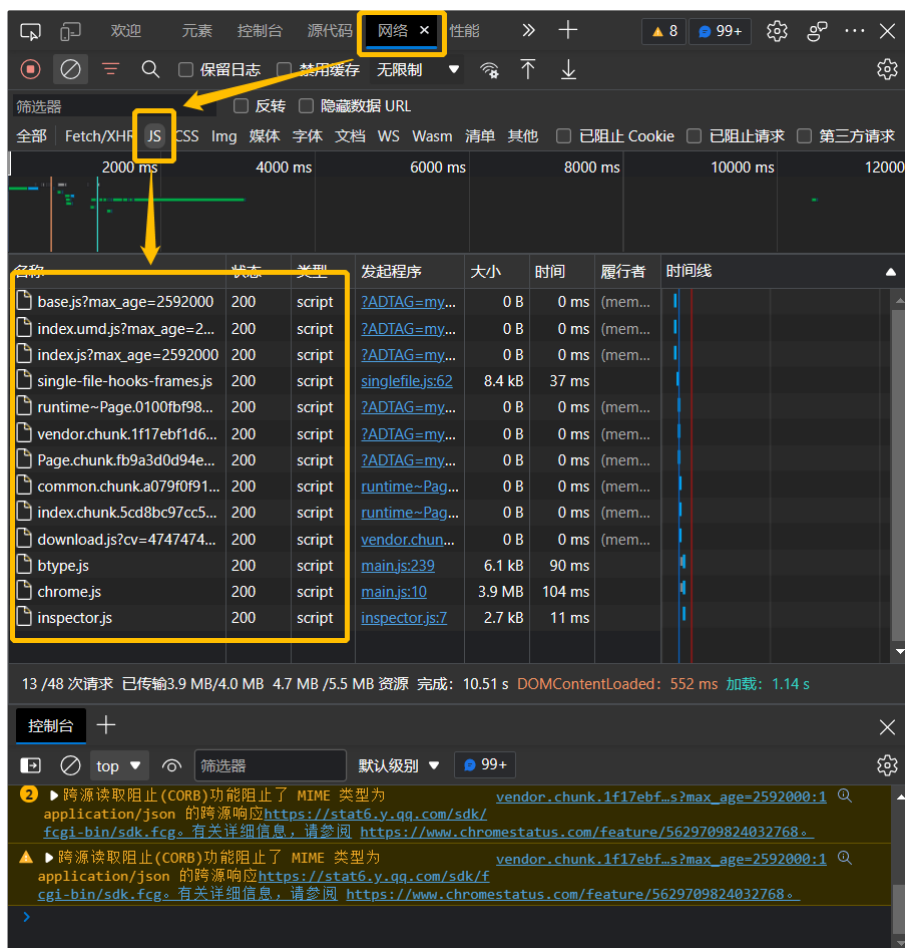


图-7 网页中使用 JavaScript 代码编写的文件

浏览器打开网页时，会先请求 HTML 文件，获得响应后，加载 HTML 的内容。当检测到有 JS 文件时，浏览器就会去请求后缀名为 .js 的文件。获得响应消息后，便会执行该文件中的 JavaScript 代码，即向 HTML 中添加内容，得到完整的页面。而当我们使用 requests 模块请求 HTML 内容时，只能获得 HTML 代码，不会进一步请求 JavaScript 文件。因此，我们获得的html代码与网页中看到的不一樣。

1.10 JSON

JSON的全称为JavaScript Object Notation，是一种用于存储和传输数据的格式。JSON是一种轻量级的文本数据交换格式，与xml类似，但比XML 更小、更快、更易理解和解析。简单来说，JSON就像字符串、列表、元组一样，用来存储和传输数据。JSON数据与字典类似，是由键值组成的，每一对键值都是以“:”进行连接，键必须是字符串并由双引号包围，引号中填写具体内容，数据与数据之间以逗号分隔。

```
1 var JSONObj = {"name": "MXP", "age": 20, "gender": true, "height": 1.75, "skill": ["Python", "C++"]};
```

上面代码创建了一个JSON对象，对象中的name就是键，MXP就是值，其他的年龄、性别、身高信息也是键值对的形式。并且JSON数据类型可以有字符串类型、整型、布尔型、浮点型以及列表等。JSON中的列表表示数组，列表中每一项以英文逗号（，）进行分割，所以数组中的每项也以英文逗号（，）进行分割，数组中每项以大括号{}包围起来的对象。

在Python中，`json.loads()` 函数能够对数据进行解码，将JSON数据与Python的数据类型进行转化，目的是能在Python中处理JSON数据。使用 `json.loads()` 函数时，将JSON数据传入括号中就能将字符串类型数据转换成字典类型的数据。解码成的数据类型需要看JSON对应的格式，格式为字典就会转成字典，为列表就会转成列表。

1.11 URL编码

互联网规范规定URL只能按照百分号加英文字母、阿拉伯数字和某些标点符号来展示，但是因为浏览器能够按照一定规则对非规定的字符进行转码，因此我们在网页端能看到中文字符。



图-8 浏览器转码

当URL中出现指定字符之外的内容，例如汉字时，就会对汉字进行转码，URL编码会使用“%”+数字或字母来替换汉字，如图所示，京东巧克力商品链接中，巧克力三个字是汉字，所以每个汉字都被%的格式替换了。

https://search.jd.com/Search?
keyword=%E5%B7%A7%E5%85%8B%E5%8A%9B
巧 克 力

图-9 汉字转码

1.12 Cookie

在我们浏览网站时，某些网站的服务器会给用户浏览器发送一小块数据，用于辨别用户身份，这个就是 Cookie。Cookie 的作用是保存用户的浏览数据，对用户身份起到辨识作用，某些网站也会用 Cookie 来识别某个请求是否为爬虫。比如，我们在网页端登录了某个网站，那么该网站就会发送 Cookie 到我们的浏览器，用于记录我们的登录状态。Cookie 主要存放在浏览器的 Request Headers 中。

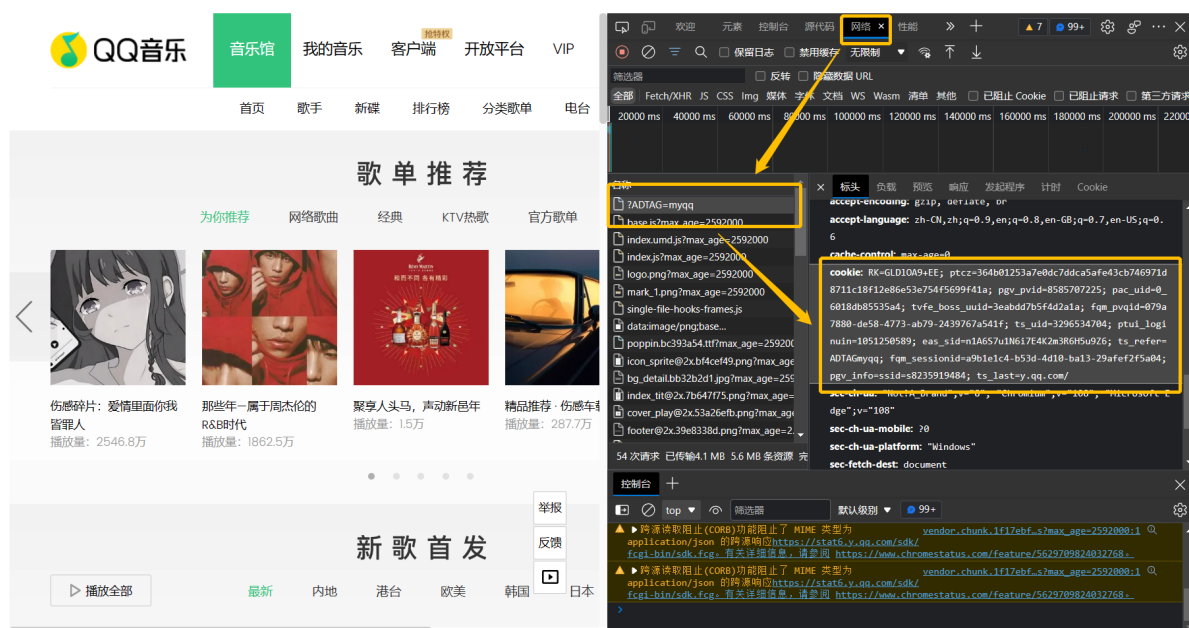


图-10 网页中的 Cookie 信息

Cookie 存放在请求头中，用于记录用户登录信息、浏览记录等。当服务器首次收到浏览器的请求时，服务器会在返回的响应消息中添加一个 Set-Cookie 数据。浏览器收到并保存该数据，之后的每次请求都会携带 Cookie 数据表明身份。

在Python代码里，可以在请求头中设置 Cookie，在请求头中复制 cookie，粘贴到代码中，以字典 key-value 的形式赋值给字典 headers。某些网站在反爬虫时，也会检查请求头中的 Cookie 或其他信息，但比较常见的是检查 User-Agent。

2. 技术细节

爬虫流程：

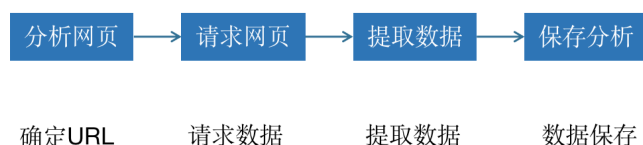


图-11 网络爬虫基本流程

2.1 requests模块

使用Python来爬取网页内容，需要安装requests模块，该模块可以用于获取网络数据。由于requests模块是Python的第三方模块，需要额外安装，安装requests模块需要在电脑终端输入代码：

```
1 | pip install requests
```

如果在电脑上安装不上或安装缓慢，可在命令后面添加如下配置进行加速：

```
1 | pip install requests -i  
https://mirrors.aliyun.com/pypi/simple/
```

安装之后，需要用import导入requests模块。

```
1 | import requests
```

对于爬虫来说，要获取网页中的内容，就需要网页的URL，打开网页，点击链接框，右键选择复制即可复制链接。requests.get()函数可用于模拟浏览器请求网页的过程，在Python语言中使用该函数，就能够获取网页数据。get()函数中传入要访问网页的URL，就像浏览器打开URL一样。

```
1 | response = requests.get(网页URL)
```

`requests.get()`是获取网页信息的主要函数，使用该函数获取案例网页的URL，会返回一个Response对象，也就是响应消息。使用`print`输出响应消息会得到`<Response [200]>`，表示响应消息中状态码为200，说明浏览器的请求执行成功。在浏览器中查看Response Headers中的信息就能够找到`status: 200`，状态码200代表请求执行成功。使用`.status_code`属性就可以查看状态码，这里输出的状态码数据类型是整型。

```
1 | print(response.status_code)
```

通过请求URL，已经获取到了Web服务器返回的信息，这些信息的呈现要用到`.text`属性，该属性能够将获取到的网页信息提取出来。

```
1 | print(response.text)
```

2.2 解析模块Beautiful Soup

对于一个网页的节点来说，它可以定义id、class或其他属性，而且节点之间还有层级关系。我们可以借助网页节点的结构和属性，提取想要的信息。在这里，可以借助一个强大的解析工具—**BeautifulSoup**。在前面的内容中提及到了**标签**，例如：`<h1>`标签、`<p>`标签。在网络爬虫中，标签也可以称为**节点**，网页中的每个部分都可以叫做节点。例如：html标签、属性、文本等都是一个节点。节点有多种类型，例如表示整个HTML文档的文档节点、表示HTML中的标签的元素节点、表示HTML标签中属性的属性节点等等。

BeautifulSoup 是 Python 的一个 HTML 或 XML^[7] 的解析模块，可以用它来从网页中提取想要的信息。BeautifulSoup不是一个内置模块，所以在使用前要先通过代码在终端中进行安装：

```
1 | pip install bs4
```

如果在电脑上安装不上或安装缓慢，可在命令后面添加如下配置进行加速：

```
1 | pip install bs4 -i https://mirrors.aliyun.com/pypi/simple/
```

网络爬虫的最终目的就是过滤选取网络信息，最重要的部分可以说是解析器。解析器的优劣决定了爬虫的速度和效率。Beautiful Soup 官方推荐我们使用的是 `lxml` 解析器，原因是它具有更高的效率，所以我们将采用`lxml`解析器。`lxml` 不是一个内置模块，所以在使用前要先通过代码在终端中进行安装：


```
1 pip install lxml
```

如果在电脑上安装不上或安装缓慢，可在命令后面添加如下配置进行加速：

```
1 pip install lxml -i https://mirrors.aliyun.com/pypi/simple/
```

安装完成后，需要使用 bs4 模块中的 BeautifulSoup 类，就要使用 `from...import` 从 bs4 中导入 BeautifulSoup 。

```
1 from bs4 import BeautifulSoup
```

BeautifulSoup() 函数可以把不标准的 HTML 代码重新进行**自动更正**，从而方便对其中的节点、标签、属性等进行操作。调用BeautifulSoup() 初始化函数，用于构建一个 BeautifulSoup 对象。在创建一个 BeautifulSoup 对象时，需要传入两个参数：第一个参数是需要解析的 HTML 文本，第二个参数是解析器的类型。

```
1 soup = BeautifulSoup(html, "lxml")
```

我们可以使用 BeautifulSoup 中的 find_all() 函数，可以根据标签名获取所有符合指定条件的节点，返回一个列表。根据标签名查询节点：

```
1 ps = soup.find_all(name = "h1")
```

示例代码中，假设要获取 h1 标签所在的节点，可以在 find_all() 中，传入 name 参数，其参数值为 h1 。将返回的结果，赋值给一个变量。输出的结果是包含所有 h1 节点的列表。

对于每一个节点，可以调用 .string 属性，用来获取节点中的内容。对于节点 <h1>高八度</h1>，调用 .string 属性 就能获取 <h1> 节点中的内容：高八度。但由于 find_all() 返回的是一个列表，我们不能直接调用 .string 属性，而是需要使用 **for 循环遍历列表**，获取每一个节点字符串，再来调用 .string 属性获取节点中的标签里的内容。需要注意的是 .string 属性只能提取单个节点或节点统一的内容，例如：<p>高八度 这是一句话</p>，p 节点包含一个子节点 em ，由于 p 节点有且只有一个子节点，使用 .string 属性时，会输出 em 节点的 .string 内容。而当提取节点包含多个子节点时，使用 .string 属性时，不清楚应该调用哪个节点的内容，会返回None值。

遇到节点中既包含其他节点，也有文字时，可以使用 .text 属性来提取内容。.text 属性能直接提取该节点中的所有文字，并返回字符串格式。

2.3 User-Agent

想要获取网页中的信息，首先要获取网页 HTML 代码，再把信息从中提取出来。只要向网页的服务器发送请求，服务器返回的响应就是网页 HTML 代码。有时候返回的状态码是418，418 其实是一个愚人节玩笑，含义是：当客户端给一个茶壶发送泡咖啡的请求时，茶壶就返回一个 418 错误状态码，表示“我是一个茶壶”，也意味着爬虫被发现了。

在进行 HTTP 请求的时候，除了请求指定的 url 信息之外，还会告诉服务端“我是谁”，“我支持哪些特性”。这些信息就是HTTP 请求头，其中声明“我是谁”的就是 HTTP 请求头部的 User-Agent 。如果用的是 requests 发起的请求，而不是浏览器，就比较容易被发现，所以需要伪装成浏览器来发送请求。

在网页中请求头和响应头信息保存在变量 headers 中，使用 响应消息.request.headers 就能查看请求头中的信息，使用 响应消息.headers 就能查看响应头中的信息。请求头中的 User-Agent 用来表示产生请求时浏览器的类型。查看使用 响应消息.request.headers 输出的请求头中，User-Agent 的值为 **python-requests**，说明此次请求是Python发出的并且能被服务器识别到。

伪装成浏览器发出请求，我们可以对请求头信息进行设置，请求头中有很多内容，常用的就是User-Agent。请求头信息需要以字典key-value的形式赋值给字典 headers，然后在 requests.get() 中添加参数 headers，其参数值为字典 headers 。

```
1 headers = {"User-Agent": "浏览器类型"}
2 response = requests.get(url, headers = headers)
```

我们用浏览器打开网页，就是浏览器对服务端发送的请求，如果要设置请求头信息，模拟成浏览器去访问网站，可以复制浏览器的 User-Agent 的值，粘贴到代码中，以字典key-value的形式赋值给字典 headers。

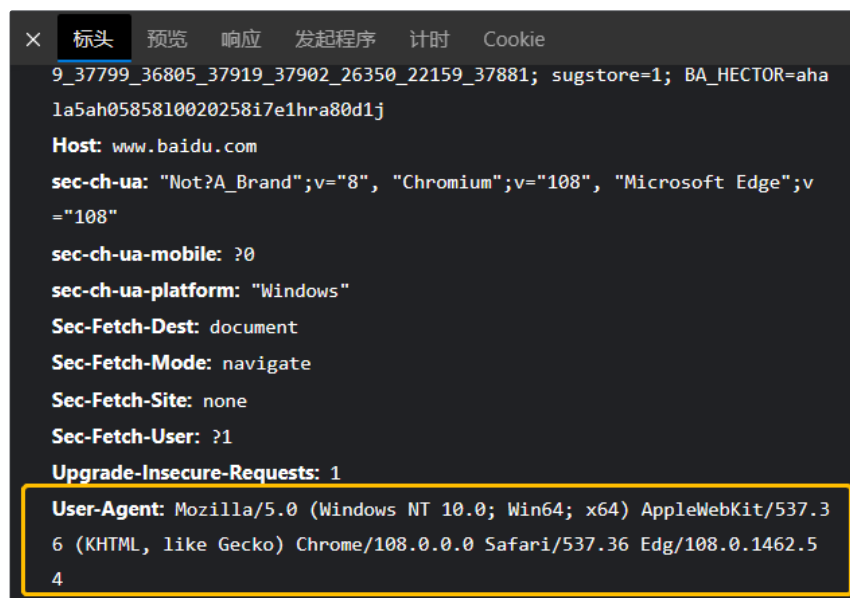


图-12 浏览器的 User-Agent 值

```
1 headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0;  
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.54"}  
2 response = requests.get(url, headers = headers)
```

2.4 按属性查询

如图-13所示，在 HTML 中，**span 标签**的作用是用来组合文档中的行内元素。**class** 属性的作用是用来给标签分类。**class="short"**也就代表着，这里标签的属性是 "short"。

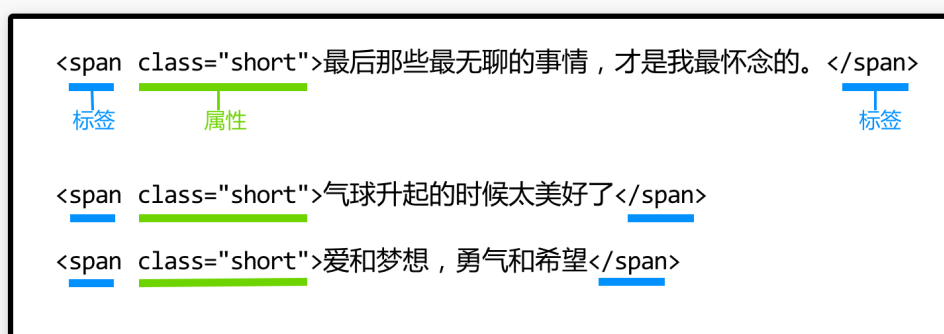


图-13 span 标签和 class 属性

在网页的检索框中，属性需要使用 (点).属性值 来定位。例如，想要定位 **class="short"** 的节点，就要在检索框中输入 **.short** 进行搜索。

在代码中，如果以属性为查询条件，就需要在 `find_all()` 中，传入 `class="short"`。由于 `class` 在 Python 里是一个关键字，所以后面需要加一个下划线，即：

```
1 content_all = soup.find_all(class_="short")
```

2.5 文件读写

在Python语言中，`open()`函数表示要打开一个文件，对文件进行处理的操作都需要用到这个函数。括号中要传入两个参数，包括要打开的文件路径，以及打开文件的方式。文件路径要使用双引号包围，双引号中就是要打开的文件路径。打开方式也要使用双引号包围，双引号中就是要对文档进行的操作。常见的文件打开方式主要有：`r`，当打开方式设置为`r`时，表示只能对该文件进行读数据操作；`w`，当打开方式设置为`w`时，表示只能对该文件进行写入操作。

表-5 常见的文件打开方式

参数	描述
r	以只读方式打开文件，文件的指针将会放在文件的开头，这是默认模式。
w	打开一个文件只用于写入，如果该文件已存在则打开文件，并从头开始编辑，即原有内容会被删除，如果该文件不存在，则创建新文件。
a	打开一个文件用于追加，如果该文件已存在，文件指针将会放在文件的结尾，也就是说，新的内容将会被写入到已有内容之后，如果该文件不存在，创建新文件进行写入。
wb	以二进制格式打开一个文件只用于写入，如果该文件已存在则打开文件，并从头开始编辑，即原有内容会被删除，如果该文件不存在，则创建新文件。

例如，使用`open()`函数以只读的方式，打开名字为`products`的`txt`文件。

```
1 f = open("/Users/products.txt", "r")      # macOS系统路径
2 f = open(r"D:\products.txt", "r")        # Windows系统路径
```

使用`read()`函数，就能够读该文件中的内容。为了节约系统资源和保存对文件的操作，对文件操作完毕后要使用`close()`关闭文件。

```
1 f = open(r"D:\products.txt", "r")
2 print(f.read())
3 f.close()
```

内容写入时，使用open()函数打开文件时，要将打开方式设置为"w"，"w"的作用就是对文件进行写入操作，如果该文件已存在则覆盖，不存在就创建新文件再写入。

使用open()函数打开文件操作完成之后，都需要使用close()来关闭文件，每次都这样有点繁琐。可以使用with语句打开文件，with语句内的代码块执行完毕后会自动调用文件的close()函数。with语句配合as关键字，可以将打开的文件赋值给变量f，便于调用函数读文件。具体写法如下：

```
1 with open(r"D:\products.txt", 'r') as f:
2     print(f.read())
```

2.6 爬取图片并保存

假设网页链接是：<https://npbcz.wordpress.com/2020/08/20/photo/>，要从网页中获取一张图片并保存到文件夹中。首先在图片上点击右键选择复制图片地址，这样就获得了图片地址：<https://npbcz.files.wordpress.com/2020/09/2-2.jpg>。通过链接获取网页中的图片数据，需要用到.content属性，使用该属性获取到的数据类型为bytes类型，表示二进制数据。with...as语句中图片写入时，需要将打开方式设置为"wb"，它可以用于图片的写入，并且如果该文件已存在则将其覆盖。如果该文件不存在，创建新文件。

```
1 import requests
2 url = "https://npbcz.files.wordpress.com/2020/09/2-2.jpg"
3 response = requests.get(url)
4 img = response.content
5 with open("/Users/图片.jpg", 'wb') as f:
6     f.write(img)
```

网页中表示图片的节点形如：

```

```

例如：

由尖括号包围的是标签，网页中的表示图片标签。在HTML中，标签属于单独标签，所以它是没有结束标签的。src是img标签中的属性，指向要展示图片的地址。"图片地址"是属性值，一般图片地址会使用URL，这样访问URL就能打开图片。alt是img标签中的属性，这个属性可选填，可用于描述图片。"图片介绍"是属性值，一般用于描述图片的内容。

接下来是获取豆瓣电影评分最高的前25部电影的海报图片并下载到本地的步骤。
(电影海报可作为电影插曲的音乐专辑封面)

在浏览器中打开网页：<https://movie.douban.com/top250>。首先来分析网页结构，鼠标放到网页中，点击右键选择检查，就能看到网页源代码，点击【选择器】（图中圈起来的小箭头），将鼠标移至第一张图片的位置，就能够看到该图片对应的源代码，如图-14 所示。

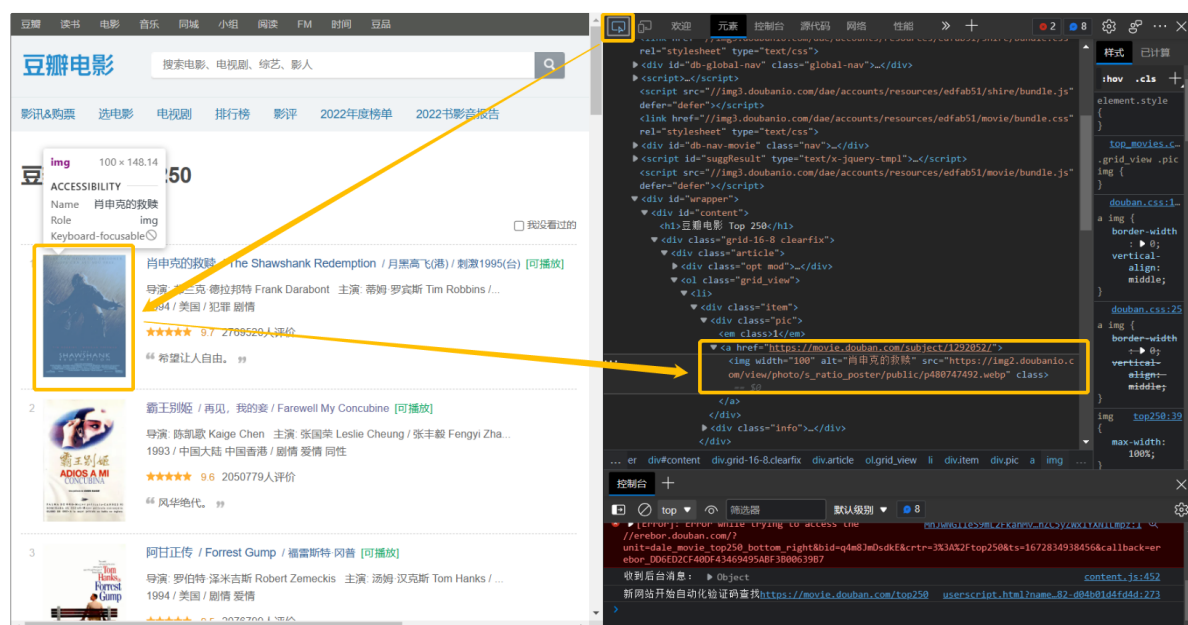


图-14 图片来源代码定位

根据前面讲到的img图片标签的格式，表示图片地址就是src所对应的内容，表示图片名字的就是alt对应的内容。接下来，使用requests模块和BeautifulSoup模块请求并解析网页内容。

```

1 import requests
2 from bs4 import BeautifulSoup
3
4 url = "https://y.qq.com/?ADTAG=myqq#type=index"
5 headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.54"}
6
7 response = requests.get(url, headers = headers)
8 html = response.text
9 soup = BeautifulSoup(html, "lxml")

```

观察网页结构，可以看到所有的图片都包含在class="pic"的属性中。所以，需要使用find_all()函数找到所有class="pic"的节点。

```

1 content_all = soup.find_all(class_ = "pic")

```

这样我们就获得了连接<https://movie.douban.com/top250>对应网页的25张海报的源代码，观察每张图片的源代码中都包含了图片标签。接下来，需要将图片标签中的URL提取出来。由于find_all()返回的是一个列表，不能直接对整个列表进行处理，而是使用for循环遍历列表content_all。通过遍历，就拿到了每张图片的源代码，这里的每个URL都放在标签中。

```

<div class="pic">
<em class="">1</em>
<a href="https://movie.douban.com/subject/1292852/">

</a>
</div>
<div class="pic">
<em class="">2</em>
<a href="https://movie.douban.com/subject/1291546/">

</a>
</div>

```

图-15 各个节点的组成

之前的find_all()函数能够匹配所有满足条件的元素，并且返回一个列表。而find()函数只返回第一个匹配到的元素，find()函数的用法与find_all()函数基本类似。每个图片对应的div元素内，只有一个img标签，要获取该标签，使用find()函数，将name="img"传入函数中，就能够拿到img对应的标签内容。这样我们就获取到了25个海报对应的图片标签，距离URL还差一步。

可以看到所有的图片链接都在src对应的属性值中，所以需要从img标签中取出src属性对应的属性值。对标签中的src和alt属性使用.attrs，就可以获取属性值。例如获取图片标签中的图片描述就可以使用，attrs["alt"]，获取图片标签中的链接就可以使用attrs["src"]。在保存图片时，需要用到图片名字，而img标签中的alt属性刚好就是每张图片的描述。至此，即可拿到所有海报图片的URL。下一步，就是获取海报图片并保存到本地。获取图片需要请求网页数据，拿到响应数据后，利用文件读写保存图片。

获取到海报的URL后，接下来使用get()函数请求图片的链接，利用.content属性获取响应消息中的图片数据。拿到图片数据就可以使用with...as语句配合open()函数保存文件：

```
1 for content in content_all:
2     imgContent = content.find("img")
3     imgUrl = imgContent.attrs["src"]
4     imgName = imgContent.attrs["alt"]
5     imgResponse = requests.get(imgUrl)
6     img = imgResponse.content
7     with open(rf"images\{imgName}.jpg", "wb") as f:
8         f.write(img)
```

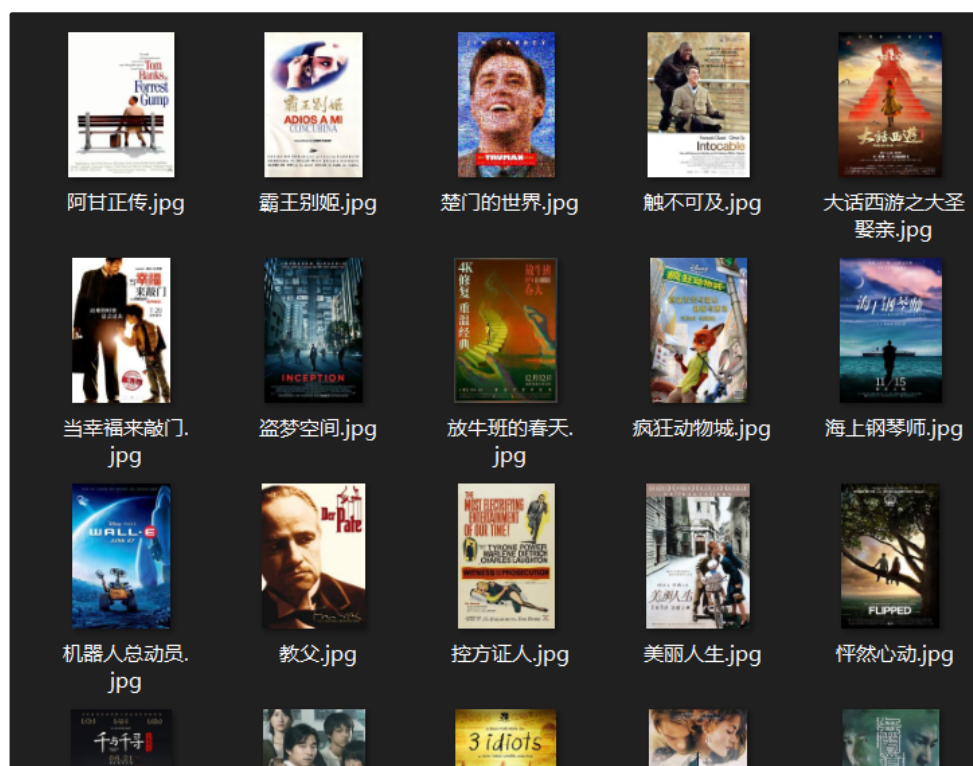


图-16 已获取且保存到本地的图片

观察上面操作获得的图片，可以发现图片的清晰度不够。放大之后比较模糊，这样的图叫做**缩略图**^[8]，为了找到高清图片，需要对网页结构进行分析，找到高清图对应的链接。

接下来，回到连接<https://movie.douban.com/top250>的对应网页，使用选择器，找到图片链接在网页中打开，点击海报图片跳转到新的页面，使用选择器，找到图片链接在新的页面打开，比较两个图片的大小，发现是一样的，即找到的两个链接相同，图片尺寸相同，并不是我们要找的高清图，要继续探索其他的页面，查看是否能够找到清晰度更高的图片。点击页面中的海报图片，使用选择器找到第一张图片的链接，复制链接在新页面打开，对比打开的图片，发现这是清晰度较高的图片。

```
1  首页图片：
2  https://img3.doubanio.com/view/photo/s_ratio_poster/public/
3  p2561716440.webp
4
5  高清图片：
6  https://img3.doubanio.com/view/photo/m/public/
7  p2561716440.webp
8
```

图-17 缩略图和高清图的链接对比

观察缩略图的链接和高清图的链接，两个链接非常相似，只要将图中标黄的部分替换成m，就变成了高清图片链接，这样就找到了缩略图转换成高清图的方法。图片的后缀名为.webp，这是Google开发的一种图片格式，网站可以使用 WebP 创建尺寸更小、细节更丰富的图片。

现在只需要使用replace()函数，将链接中的s_ratio_poster替换成m。替换后输出的链接中图片后缀名为.jpg，原因是豆瓣的服务器会根据请求信息判断要返回的图片类型，默认是.jpg。接下来，重新请求图片链接，使用with语句配合open()函数写入图片，就能够获得该页所有海报的高清图片：

```
1  for content in content_all:
2      imgContent = content.find("img")
3      imgUrl = imgContent.attrs["src"]
4      imgName = imgContent.attrs["alt"]
5      imgUrlHd = imgUrl.replace("s_ratio_poster", "m")
6      imgResponse = requests.get(imgUrl)
7      img = imgResponse.content
8      with open(rf"images\{imgName}.jpg", "wb") as f:
9          f.write(img)
```

观察网页URL的变化规律，每次翻页start=参数增加25。所以，这里可以利用for循环生成不同页面的链接。链接：<https://movie.douban.com/top250?start=0&filter=>翻页过程中，链接的大部分内容都不改变，只有start=后面的内容会变化，每次翻页参数增加25。所以，如果要查看10页的电影海报，那么就可以用for循环和range()函数，生成每次翻页的参数变化。利用for循环生成了每个链接中变化的参数后，利用字符串拼接，将参数前面内容、参数和参数后面内容，三部分拼接起来即可。需要注意的是，for循环生成的参数属于整型，不能直接与字符串进行拼接，需要将**整型转为字符串**再处理。

```
1 for i in range(0, 11):
2     page = i * 25
3     url = "https://movie.douban.com/top250?start=" +
        str(page) + "&filter="
```

通过对for循环和整数转字符串知识点的组合，就生成了多个URL。接下来，将之前单个页面的URL替换成可生成多个URL的代码，即可批量获取多页网页的图片。

2.7 编码属性获取

可以使用相关属性获取网页编码和 requests 模块编码的编码方式。requests.get()请求发出后，requests 模块会基于 HTTP 头部作出推测。在使用 .text 属性前，requests 模块会使用推测出来的文本编码，我们可以使用 .encoding 属性，找出 requests 模块使用了什么编码。而 .apparent_encoding 属性则会从网页的内容中分析网页编码的方式。

表-6 2个属性的对比

属性	说明
.encoding	从HTTP header中猜测的响应内容编码方式
.apparent_encoding	从内容中分析出的响应内容编码方式（备选编码方式）

当使用 .text 属性获取网页内容的时候，使用的是 .encoding 进行编码，但是基于 HTTP 头部分析出来的编码不一定正确，就有可能造成乱码。.apparent_encoding 是通过内容分析出编码，有一定准确率。遇到乱码的时候，可以作为爬虫备选的编码方式。当出现乱码的情况时，使用 response.encoding 属性就可以修改编码，即将网站的编码方式 response.apparent_encoding 赋值给 response.encoding。

```
1 | response.encoding = response.apparent_encoding
```

2.8 查找JS文件

JS 文件是需要请求加载的，如果不点击查看，网页就不会加载相应的信息。如果想要查看 JS 文件，首先要点击 Network，然后选择 JS，然后点击网页中的链接加载相应的 JS 文件，就能查看目前加载的 JS 文件。

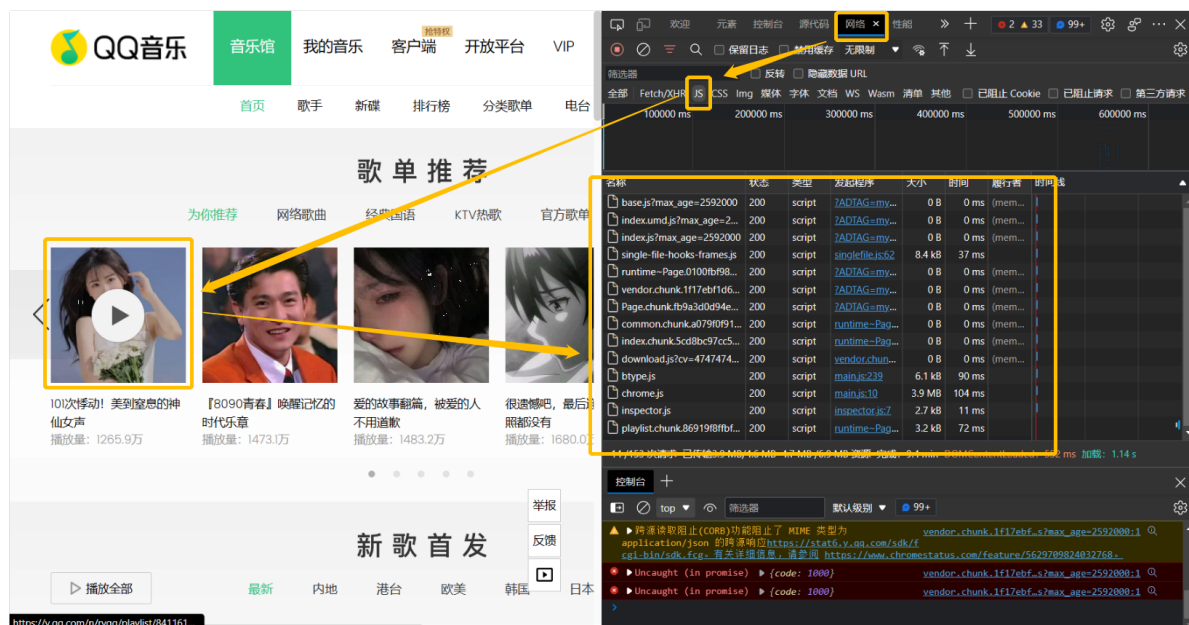


图-18 JS 文件查找

如果要查看 .js 文件中的内容，则需要逐个点开 JS 文件，选择右侧的 Preview 预览页面，查看预览代码（如果有三角折叠的信息，也要展开查看）。

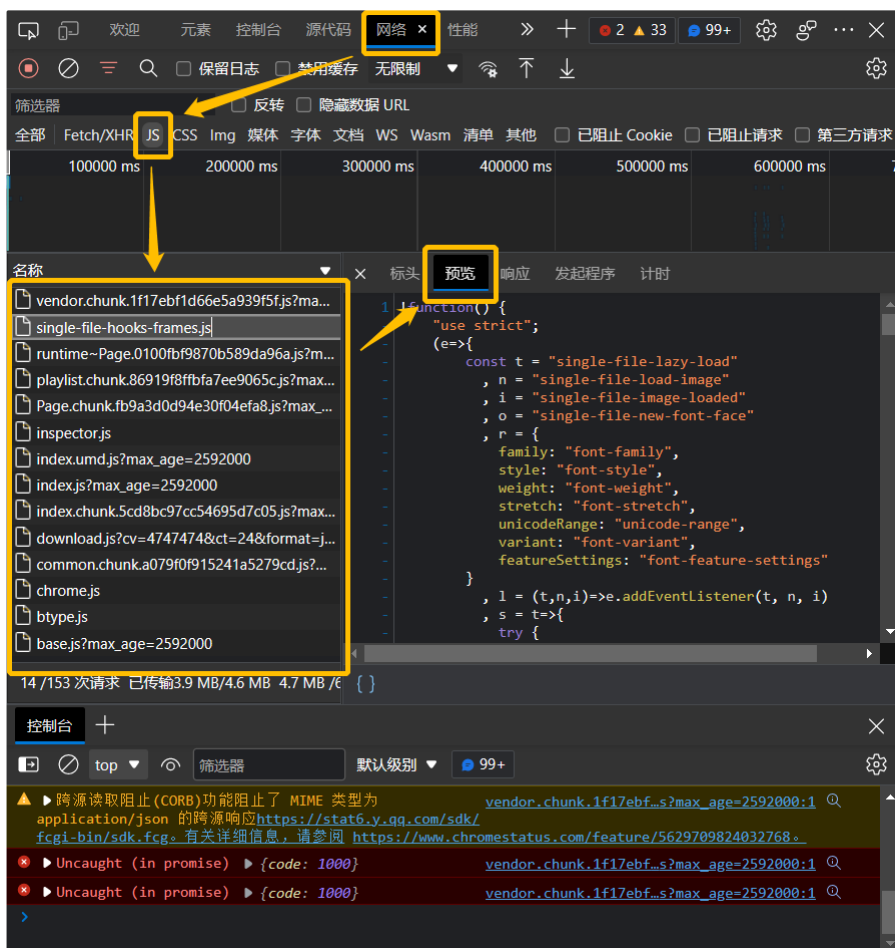


图-19 .js 文件内容查看

2.9 节点选择器

前面的内容中，提取内容时使用的 `find()` 和 `find_all()` 函数，是 BeautifulSoup 提供的**方法选择器**，就是查询符合条件的元素时，传入属性或标签名称，就可以得到符合条件的元素。此外，BeautifulSoup 还提供了另一种方法——**节点选择器**，通过调用节点的名称查询元素。

使用节点选择器定位节点的方法如下：

```

1 | html = """
2 | <title> Example </title>
3 | <p class="title" name="rank">
4 |     <em>first</em>
5 |     <em>second</em>
6 | </p>
7 | """
8 | from bs4 import BeautifulSoup
9 | soup = BeautifulSoup(html, "lxml")
10 | content = soup.title

```

节点选择器直接调用节点的名称就能定位到节点。示例的 html 代码中要想定位到 <title></title> 节点, 就可以使用 `soup.title` 来表示。

使用节点选择器获取节点名称的方法如下:

```

1 | content = soup.title.name

```

调用节点的名称时, 使用 `.name` 属性就能定位到名称。示例的 html 代码中要想提取 <title></title> 节点的名称, 就可以使用 `soup.title.name` 来表示。

使用节点选择器获取节点属性的方法如下:

```

1 | content = soup.p.attrs
2 | content = soup.p.attrs["class"]

```

调用节点时, 使用 `.attrs` 就能定位到节点的属性。如果节点中有多个属性, 就会全部输出。示例的 html 代码中要想提取 <p></p> 节点中所有的属性, 就可以使用 `soup.p.attrs` 来表示。<p> 节点中有两个属性, 以字典的形式输出。

如果只提取某个属性的属性值, 可以使用 `.attrs[属性]`。示例的 html 代码中要想提取 <p></p> 节点中 `class` 的属性值, 就可以使用 `soup.p.attrs["class"]` 来表示。想要提取 <p> 节点中 `name` 的属性值, 使用 `soup.p.attrs["name"]` 来表示。

使用节点选择器获取节点内容的方法如下:

```

1 | content = soup.title.string
2 | content = soup.p.string # 结果为 None

```

调用节点时，使用 `.string` 属性就能定位到节点的内容。示例的 `html` 代码中要想提取 `<title></title>` 节点中的内容，就可以使用 `soup.title.string` 来表示。当该节点下同时包含了多个子节点时，就无法确定 `.string` 应该调用哪个子节点的内容，输出结果就是 `None`。例如，提取 `html` 中 `p` 节点的内容时，使用 `soup.p.string` 就会输出 `None`。因为 `p` 节点中有两个 `em` 节点，使用 `.string` 时，不清楚提取哪个内容就会返回 `None`。

当节点下同时包含了多个子节点时，需要使用 `.text` 属性提取节点中的全部内容：

```
1 content = soup.p.text
```

例如，要提取 `html` 中 `p` 节点的全部内容，可以使用 `soup.p.text` 即可。

当节点下同时包含了多个子节点时，也可以定位到下一级节点，提取该节点中的内容：

```
1 content = soup.p.em.string
```

例如，要提取 `html` 中 `p` 节点中第一个 `em` 节点的内容，可以使用 `soup.p.em.string` 即可。使用节点选择器时，在某个节点中同时存在两个相同的节点时，节点选择器只能调用第一个出现的节点，代码中第二个 `em` 标签中的内容就不会输出。

2.10 获取子结点

```
1 from bs4 import BeautifulSoup
2 html = '''
3 <span class="price" style="font-size:28px">
4   <em>¥</em>169.5</span>
5   '''
6 soup = BeautifulSoup(html, "lxml")
7 title = soup.find(class_="price")
8 child = title.contents
```

在做选取的时候，有时候不能做到一步就选到想要的节点元素，需要先选中某一个节点元素，然后以它为准再选择它的子节点等。想要获取子节点，可以使用 `.contents` 属性，得到的结果是包含子节点的列表。

3. 项目应用示例 —— 爬取网易云音乐

首先打开网易云音乐排行榜的网页，然后利用网页“F12”的搜索功能找到歌曲对应的源代码信息，如图-20所示：

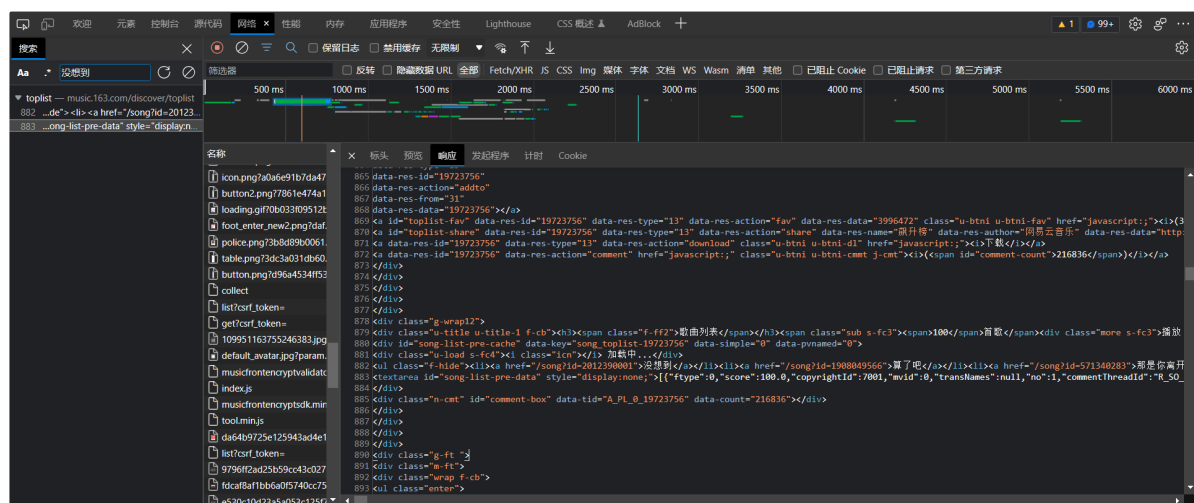


图-20 排行榜歌曲对应源代码信息

从“标头”中找到数据来源链接：

<https://music.163.com/discover/toplist>，然后就可以利用此链接在代码中发送请求，并输出状态码查看是否请求成功。

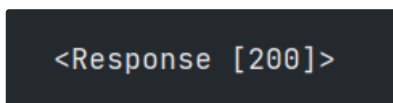


图-21 状态码为200，表示请求成功

确定请求成功后，先利用 BeautifulSoup 模块解析源代码。源代码解析后，可以发现，包含歌曲信息节点处于属性为 "f-hide" 的节点当中，且结构为：

```
<li><a href="/song/?id=.*?">.*?</a></li>
```

了解了歌曲信息节点的特点后，就可以借助 find_all() 函数来找到对应歌曲信息的节点，并输出：

```
<li><a href="/song?id=2012390001">没想到</a></li>
<li><a href="/song?id=1908049566">算了吧</a></li>
<li><a href="/song?id=571340283">那是你离开了北京的生活</a></li>
<li><a href="/song?id=2012738440">守护</a></li>
<li><a href="/song?id=2009744753">活死人2022没有Cypher</a></li>
<li><a href="/song?id=1958557540">golden hour</a></li>
<li><a href="/song?id=2012487919">刀麻发鬓角</a></li>
<li><a href="/song?id=1993067247">月下</a></li>
```

图-22 包含歌曲信息的节点（部分）

继续观察信息的特点，发现不同歌曲之间，差异在于链接的 id 以及歌曲的名字，因此可以先把歌曲的 id 和名字在代码中先提取出来，并输出查看：

```
没想到 2012390001
算了吧 1908049566
那是你离开了北京的生活 571340283
守护 2012738440
活死人2022没有Cypher 2009744753
golden hour 1958557540
刀麻发鬓角 2012487919
月下 1993067247
```

图-23 歌曲名字以及对应Id

利用所得到的 id 拼接成请求下载音乐的url: <http://music.163.com/song/media/outer/url?id={musicId}>，得到url后，即可发送请求得到每首音乐的二进制信息并保存到本地。

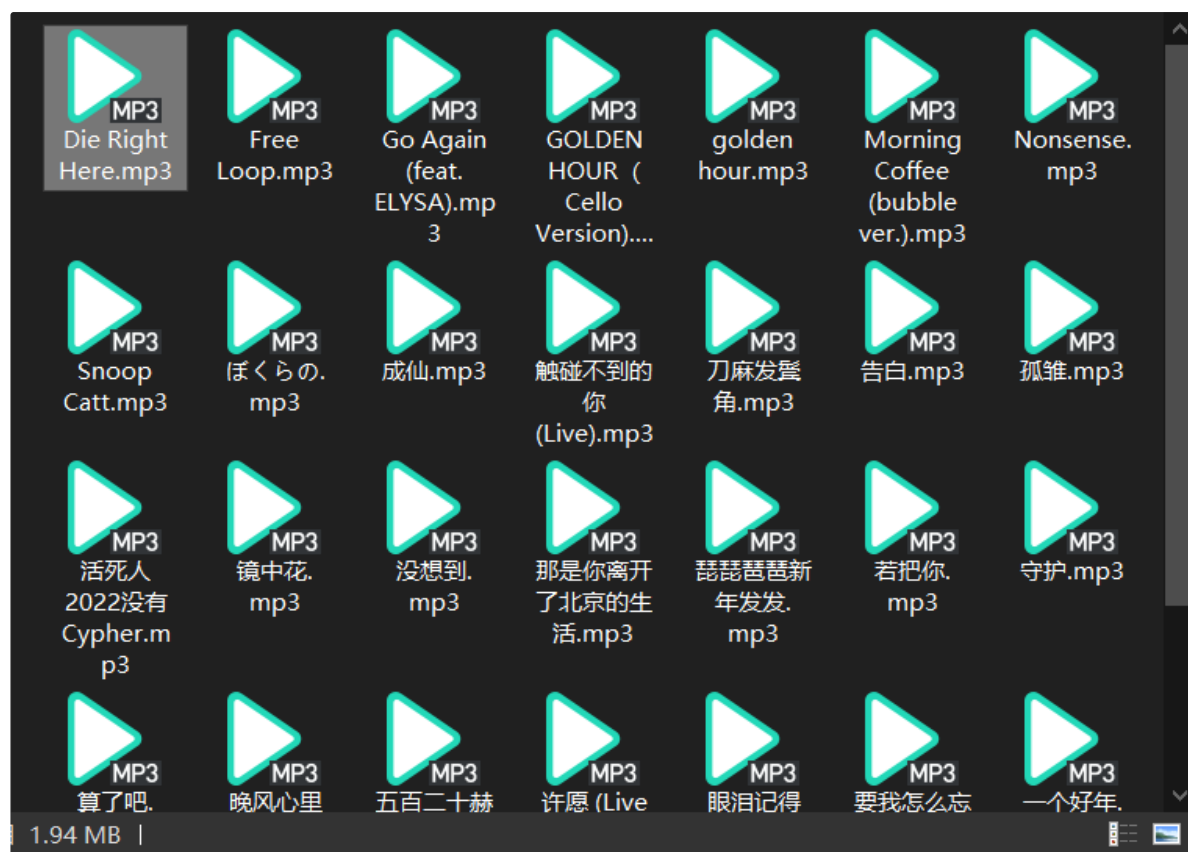


图-24 成功保存到本地的音乐（部分）

详细代码如下：

```
1 # 爬取网易云排行版音乐
2 import requests
3 from bs4 import BeautifulSoup
4 import time
```

```

5
6 url = "https://music.163.com/discover/toplist"
7 headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.76"}
8
9 response = requests.get(url, headers = headers)
10 html = response.text
11 soup = BeautifulSoup(html, "lxml")
12 content_all = soup.find_all(class_ = "f-hide")
13 content_all = content_all[0].find_all("li")
14
15 for content in content_all:
16     time.sleep(1.5)
17     musicId = content.a.attrs["href"].strip("/song?id=")
18     musicName = content.a.string
19     musicUrl = f"http://music.163.com/song/media/outer/url?
id={musicId}"
20     musicResponse = requests.get(musicUrl, headers =
headers)
21     music = musicResponse.content
22     with open(rf"musics\{musicName}.mp3", "wb") as file:
23         file.write(music)

```

-
1. 源代码 (Source Code) 是指人类能够看得懂的计算机语言指令，被翻译成计算机可执行的代码。 ↩
 2. 如果在自己的电脑上使用浏览器浏览网页，那么这里的客户端就是浏览器。 ↩
 3. 浏览网页时服务器的作用就是将网页信息提供给浏览器，此时的服务器也被称为Web服务器。 ↩
 4. 互联网中有多种协议，比如FTP、SFTP、SMB等。 ↩
 5. 安全保护就是SSL协议，SSL (Secure Socket Layer) ，既安全套接字协议，是独立于HTTP的协议，可以说SSL是当今世界上应用最广泛的网络安全技术。 ↩
 6. 每一台访问互联网的设备都有的一个网络地址，有时多台设备会共享同一个地址去访问互联网上的资源（比如同一个家庭路由器下的所有设备）。 ↩
 7. XML设计用来传送及携带数据信息，不用来表现或展示数据，HTML则用来表现数据，所以XML用途的焦点是它说明数据是什么，以及携带数据信息。 ↩
 8. 缩略图是经过压缩方式处理的小图，相当于图片文件预览，在网页预览中，加载速度快。 ↩