第 1 章

理解网络爬虫

1.1 爬虫的定义

网络爬虫是一种按照一定的规则自动地抓取网络信息的程序或者脚本。简单来说，网络爬虫就是根据一定的算法实现编程开发，主要通过 URL 实现数据的抓取和发掘。

随着大数据时代的发展，数据规模越来越庞大，数据类型繁多，但是数据价值普遍较低。为了从庞大的数据体系里获取有价值的数据，从而延伸了网络爬虫、数据分析等多个职位。近几年，网络爬虫的需求更是井喷式地爆发，在招聘的供求市场上往往是供不应求，造成这个现状的主要原因就是求职者的专业水平低于需求企业的要求。

传统的爬虫有百度、Google、必应等搜索引擎，这类通用的搜索引擎都有自己的核心算法。但是，通用的搜索引擎存在着一定的局限性：

（1）不同的搜索引擎对于同一个搜索会有不同的结果，搜索出来的结果未必是用户需要的信息。

（2）通用的搜索引擎扩大了网络覆盖率，但有限的搜索引擎服务器资源与无限的网络数据资源之间的矛盾将进一步加深。

（3）随着网络上数据形式繁多和网络技术的不断发展，图片、数据库、音频、视频多媒体等不同数据大量出现，通用搜索引擎往往对这些信息含量密集且具有一定结构的数据无能为力，不能很好地发现和获取。

因此，为了得到准确的数据，定向抓取相关网页资源的聚焦爬虫应运而生。聚焦爬虫是一个自动下载网页的程序，可根据设定的抓取目标有目的性地访问互联网上的网页与相关的 URL，从而获取所需要的信息。与通用爬虫不同，聚焦爬虫并不追求全面的覆盖率，而是抓取与某一特定内容相关的网页，为面向特定的用户提供准备数据资源。

1.2 爬虫的类型

网络爬虫根据系统结构和开发技术大致可以分为 4 种类型：通用网络爬虫、聚焦网络爬虫、增量式网络爬虫和深层网络爬虫。

通用网络爬虫又称全网爬虫，常见的有百度、Google、必应等搜索引擎，爬行对象从一些初始 URL 扩充到整个网站，主要为门户站点搜索引擎和大型网站服务采集数据，具有以下特点：

（1）由于商业原因，引擎的算法是不会对外公布的。

（2）这类网络爬虫的爬取范围和数量巨大，对于爬取速度和存储空间要求较高，爬取页面的顺序要求相对较低。

（3）待刷新的页面太多，通常采用并行工作方式，但需要较长时间才能刷新一次页面。

（4）存在一定缺陷，通用网络爬虫适用于为搜索引擎搜索广泛的需求。

聚焦网络爬虫又称主题网络爬虫，是选择性地爬取根据需求的主题相关页面的网络爬虫。与通用网络爬虫相比，聚焦爬虫只需要爬取与主题相关的页面，不需要广泛地覆盖无关的网页，很好地满足一些特定人群对特定领域信息的需求。

增量式网络爬虫是指对已下载网页采取增量式更新和只爬取新产生或者已经发生变化的网页的爬虫，它能够在一定程度上保证所爬取的页面尽可能是新的页面。只会在需要的时候爬取新产生或发生更新的页面 ，并不重新下载没有发生变化的页面，可有效减少数据下载量，及时更新已爬取的网页，减小时间和空间上的耗费，但是增加了爬取算法的复杂度和实现难度，基本上这类爬虫在实际开发中不太普及。

深层网络爬虫是大部分内容不能通过静态 URL 获取的、隐藏在搜索表单后的、只有用户提交一些关键词才能获得的网络页面。例如某些网站需要用户登录或者通过提交表单实现提交数据。这类爬虫也是本书讲述的重点之一。

实际上，聚焦网络爬虫、增量式网络爬虫和深层网络爬虫可以通俗地归纳为一类，因为这类爬虫都是定向爬取数据。相比于通用爬虫，这类爬虫比较有目的性，也就是网络上经常说的网络爬虫，而通用爬虫在网络上通常称为搜索引擎。

1.3 爬虫的原理

通用网络爬虫的实现原理及过程如图 1-1 所示。

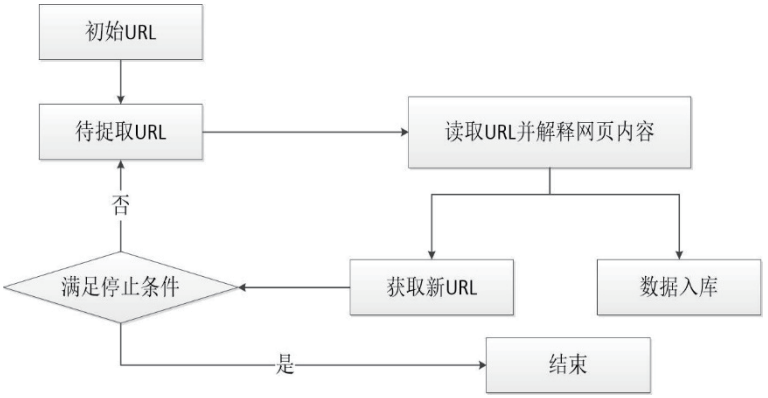


图 1-1　通用爬虫实现的原理及过程

通用网络爬虫的实现原理：

（1）获取初始的 URL。初始的 URL 地址可以人为地指定，也可以由用户指定的某个或某几个初始爬取网页决定。

（2）根据初始的 URL 爬取页面并获得新的 URL。获得初始的 URL 地址之后，先爬取当前 URL 地址中的网页信息，然后解析网页信息内容，将网页存储到原始数据库中，并且在当前获得的网页信息里发现新的 URL 地址，存放于一个 URL 队列里面。

（3）从 URL 队列中读取新的 URL，从而获得新的网页信息，同时在新网页中获取新 URL，并重复上述的爬取过程。

（4）满足爬虫系统设置的停止条件时，停止爬取。在编写爬虫的时候，一般会设置相应的停止条件，爬虫则会在停止条件满足时停止爬取。如果没有设置停止条件，爬虫就会一直爬取下去，一直到无法获取新的 URL 地址为止。

聚焦网络爬虫的执行原理和过程与通用爬虫大致相同，在通用爬虫的基础上增加两个步骤：定义爬取目标和筛选过滤 URL，原理如图 1-2 所示。

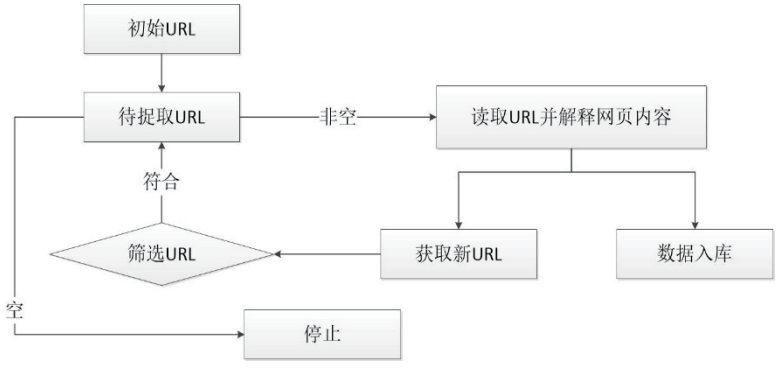


图 1-2　聚焦网络爬虫的原理

聚焦网络爬虫的实现原理：

（1）制定爬取方案。在聚焦网络爬虫中，首先要依据需求定义聚焦网络爬虫爬取的目标以及整体的爬取方案。

（2）设定初始的 URL。

（3）根据初始的 URL 抓取页面，并获得新的 URL。

（4）从新的 URL 中过滤掉与需求无关的 URL，将过滤后的 URL 放到 URL 队列中。

（5）在 URL 队列中，根据搜索算法确定 URL 的优先级，并确定下一步要爬取的 URL 地址。因为聚焦网络爬虫具有目的性，所以 URL 的爬取顺序不同会导致爬虫的执行效率不同。

（6）得到新的 URL，将新的 URL 重现上述爬取过程。

（7）满足系统中设置的停止条件或无法获取新的 URL 地址时，停止爬行。

1.4 爬虫的搜索策略

在互联网数据时代，有三大搜索策略需要有所了解，下面一一介绍。

1. 深度优先搜索

深度优先搜索是在开发爬虫早期使用较多的方法，目的是达到被搜索结构的叶结点（那些不包含任何超级 URL 的 HTML 文件）。在一个 HTML 文件中，当一个 URL 被选择后，被选 URL 将执行深度优先搜索，搜索后得到新的 HTML 文件，再从新的 HTML 获取新的 URL 进行搜索，以此类推，不断地爬取 HTML 中的 URL，直到 HTML 中没有 URL 为止。

深度优先搜索沿着 HTML 文件中的 URL 走到不能再深入为止，然后返回到某一个 HTML 文件，再继续选择该 HTML 文件中的其他 URL。当不再有其他 URL 可选择时，说明搜索已经结束。其优点是能遍历一个 Web 站点或深层嵌套的文档集合。缺点是因为 Web 结构相当深，有可能造成一旦进去再也出不来的情况发生。

举个例子，比如一个网站的首页里面带有很多 URL，深度优先通过首页的 URL 进入新的页面，然后通过这个页面里的 URL 再进入新的 URL，不断地循环下去，直到返回的页面没有 URL 为止。如果首页有两个 URL，选择第一个 URL 后，生成新的页面就不会返回首页，而是在新的页面选择一个新的 URL，这样不停地访问下去。

2. 宽度优先搜索

宽度优先搜索是搜索完一个 Web 页面中所有的 URL，然后继续搜索下一层，直到底层为止。例如，首页中有 3 个 URL，爬虫会选择其中之一，处理相应的页面之后，然后返回首页再爬取第二个 URL，处理相应的页面，最后返回首页爬取第三个 URL，处理第三个 URL 对应的页面。

一旦一层上的所有 URL 都被选择过，就可以开始在刚才处理过的页面中搜索其余的 URL，这就保证了对浅层的优先处理。当遇到一个无穷尽的深层分支时，不会导致陷进深层文档中出不来的情况发生。宽度优先搜索策略还有一个优点，能够在两个页面之间找到最短路径。

宽度优先搜索策略通常是实现爬虫的最佳策略，因为它容易实现，而且具备大多数期望的功能。但是如果要遍历一个指定的站点或者深层嵌套的 HTML 文件集，用宽度优先搜索策略就需要花费较长时间才能到达最底层。

3. 聚焦爬虫的爬行策略

聚焦爬虫的爬行策略只针对某个特定主题的页面，根据“最好优先原则”进行访问，快速、

有效地获得更多与主题相关的页面，主要通过内容与 Web 的 URL 结构指导进行页面的抓取。聚焦爬虫会给所下载的页面一个评价分，根据得分排序插入一个队列中。最好下一个搜索对弹出队列的第一个页面进行分析后执行，这种策略保证爬虫能优先跟踪那些最有可能 URL 到目标页面的页面。

决定网络爬虫搜索策略的关键是评价 URL 价值，即 URL 价值的计算方法，不同的价值评价方法计算出的 URL 的价值不同，表现出的 URL 的“重要程度”也不同，从而决定不同的搜索策略。由于 URL 包含于页面之中，而通常具有较高价值的页面包含的 URL 也具有较高价值，因此对 URL 价值的评价有时也转换为对页面价值的评价。

1.5　爬虫的合法性与开发流程

网络爬虫在大多数情况下都不会违法，在生活中几乎都有爬虫应用，比如在百度中搜索的内容几乎都是通过爬虫采集下来的，因此网络爬虫作为一门技术，技术本身是不违法的，且在大多数情况下可以放心使用爬虫技术。当然也有特殊情况，正如水果刀本身在法律上并不被禁止使用，但是用来伤害他人，这就触犯了法律规则。一般情况下，爬虫所带来的违法风险主要体现在以下几个方面：

（1）利用爬虫技术与黑客技术结合，攻击网站后台，从而窃取后台数据。因为爬虫是爬取网站上的网页信息，这些信息能给用户浏览，也就是说这些信息允许我们使用和爬取。但网站的后台数据是不被公开的数据，这些数据涉及了用户的隐私和财产安全，如果通过爬虫技术与黑客技术窃取后台数据，这就明显触发法律的底线。

（2）利用爬虫恶意攻击网站，造成网站系统的瘫痪。爬虫是通过程序去访问并操控网站，因此访问速度非常快，再加上程序的高并发处理，可以在短时间内模拟成千上万的用户在访问网站。当网站的访问量过高，就会加重网站的负载，从而造成系统的瘫痪，如果长期这样恶意攻击网站系统，也很可能违反相关的法律条例。

综上所述，爬虫技术本身是无罪的，问题往往出在人的无限欲望上。因此爬虫开发者和企业经营者的道德良知才是避免触碰法律底线的根本所在。

既然爬虫技术是合法的，那么，我们有必要了解一下爬虫的开发流程。只有掌握开发流程，才能编写高质的爬虫程序，这好比盖房子一样，建筑施工人员需要根据房屋设计图才能搭建房子，而房屋设计图等同于爬虫的开发流程。一般情况下，爬虫的开发流程如下：

（1）需求说明。任何程序开发都离不开需求说明，爬虫开发也是如此。需求说明包含功能说明、功能的业务逻辑等详细说明。爬虫的需求说明要明确告知开发人员需要爬取哪些数据、数据的存储方式以及爬虫的爬取效率。

（2）爬虫开发计划。根据爬虫的需求说明制定相关的开发计划，比如选择爬虫的开发工具、功能模块化设计、设计爬虫运行模式等一系列开发明细。

（3）爬虫的功能开发。根据开发计划编写相应的功能代码。以功能模块化设计为依据，每个功能模块以函数或类的形式表示，再将各个模块进行组合，从而实现整个爬虫功能的开发。

（4）爬虫的部署与交付。程序开发完成后（包含测试通过）就可以进行部署上线或交付客户。

部署和交付的方式有多种，比如打包 exe 程序、GUI 界面（爬虫软件）或定时执行等。

上述的爬虫开发流程是相对而言的，每一个开发步骤并非一成不变的，具体的开发流程还需要结合实际情况而定。

1.6 本 章 小 结

网络爬虫的类型理论上分为 4 类，但实际上主要是两大类：通用爬虫和聚焦爬虫。通用爬虫主要有 Google、百度、必应等搜索引擎，主要以核心算法为主导，学习成本相对较高。聚焦爬虫就是定向爬取数据，是有目的性的爬虫，学习成本相对较低。

我们常说的网络爬虫大多数以聚焦爬虫为主，其原理和过程与通用爬虫大致相同，读者在编写爬虫程序的时候，需要以设定的爬虫规则和爬取目标为主导，这样更具较强的目的性。

网络爬虫在大多数情况下都不会违法，在生活中几乎都有爬虫应用，比如在百度中搜索的内容几乎都是通过爬虫采集下来的，因此网络爬虫作为一门技术，技术本身是不违法的，且在大多数情况下可以放心使用爬虫技术。当然也有特殊情况，正如水果刀本身在法律上并不被禁止使用，但是用来伤害他人，这就触犯了法律规则。

既然爬虫技术是合法的，那么，我们有必要了解爬虫的开发流程。只有掌握开发流程，才能编写高质的爬虫程序，这好比盖房子一样，建筑施工人员需要根据房屋设计图才能搭建房子，而房屋设计图等同于爬虫的开发流程。

第 2 章

爬虫开发基础

2.1 HTTP 与 HTTPS

HTTP（Hyper Text Transfer Protocol，超文本传输协议）是一个客户端和服务器端请求和应答的标准（TCP）。客户端是终端用户，服务器端是网站。通过使用 Web 浏览器、网络爬虫或者其他工具，客户端发起一个到服务器上指定端口（默认端口为 80）的 HTTP 请求，这个客户端叫用户代理（User Agent）。响应的服务器上存储着资源，比如 HTML 文件和图像，这个服务器为源服务器（Origin Server），在用户代理和服务器中间可能存在多个中间层，比如代理、网关或者隧道（Tunnels）。

通常，由 HTTP 客户端发起一个请求，建立一个到服务器指定端口（默认是 80 端口）的 TCP 连接，HTTP 服务器则在那个端口监听客户端发送过来的请求，一旦收到请求，服务器（向客户端）发回一个状态行（比如"HTTP/1.1 200 OK"）和（响应的）消息，消息的消息体可能是请求的文件、错误消息或者其他一些信息。

在浏览器的地址栏输入的网站地址叫作 URL（Uniform Resource Locator，统一资源定位符）。就像每家每户都有一个门牌地址一样，每个网页也都有一个 Internet 地址。在浏览器的地址框中输入一个 URL 或单击一个超级 URL 时，URL 就确定了要浏览的地址，向服务器发送一次请求，浏览器通过超文本传输协议（HTTP）传送到服务器，服务器根据请求头做出相应的响应，将响应数据返回到客户端，客户端收到响应内容后，通过浏览器翻译成网页。

HTTP 协议传输的数据都是未加密的，也就是明文的数据，因此使用 HTTP 协议传输隐私信息非常不安全。为了保证这些隐私数据能加密传输，于是网景公司设计了 SSL（Secure Sockets Layer）协议用于对 HTTP 协议传输的数据进行加密，从而诞生了 HTTPS。

HTTPS（Hyper Text Transfer Protocol orer Secure Sokcket Layer，可以理解为 HTTP+SSL/TLS）在传输数据之前需要客户端（浏览器）与服务端（网站）之间进行一次握手，在握手过程中将确立双方加密传输数据的密码信息。HTTP 与 HTTPS 的主要区别可参考图 2-1 所示。

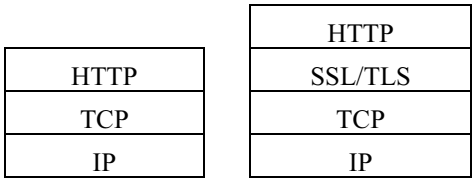


图 2-1　HTTP 与 HTTPS 的区别

HTTPS 的 SSL 中使用了非对称加密、对称加密以及 HASH 算法。握手过程的简单描述如下：

（1）浏览器将自己支持的一套加密规则发送给网站。

（2）网站从中选出一组加密算法与 HASH 算法，并将自己的身份信息以证书的形式发回给浏览器。证书里面包含网站地址、加密公钥以及证书的颁发机构等信息。

（3）获得网站证书之后浏览器要做以下工作：

① 验证证书的合法性（如颁发证书的机构是否合法、证书中包含的网站地址是否与正在访问的地址一致等），如果证书受信任，浏览器栏就会显示一个小锁头，否则会给出证书不受信任的提示。

② 如果证书受信任或者用户接受了不受信任的证书，浏览器就会生成一串随机数的密码，并用证书中提供的公钥加密。

③ 使用约定好的 HASH 计算握手消息，并使用生成的随机数对消息进行加密，最后将之前生成的所有信息发送给网站。

（4）网站接收浏览器发来的数据之后要做以下操作：

① 使用自己的私钥将信息解密并取出密码，使用密码解密浏览器发来的握手消息，并验证 HASH 是否与浏览器发来的一致。

② 使用密码加密一段握手消息，发送给浏览器。

（5）如果浏览器解密并计算握手消息的 HASH 与服务端发来的 HASH 一致，此时握手过程结束，之后所有的通信数据将使用之前浏览器生成的随机密码，并利用对称加密算法进行加密。

浏览器与网站互相发送加密的握手消息并验证，目的是保证双方都获得一致的密码，并且可以正常地加密、解密数据，为真正数据的传输做一次测试。另外，HTTPS 一般使用的加密与 HASH 算法如下。

（1）非对称加密算法：RSA、DSA/DSS。

（2）对称加密算法：AES、RC4、3DES。

（3）HASH 算法：MD5、SHA1、SHA256。

其中，非对称加密算法用于在握手过程中加密生成的密码，对称加密算法用于对真正传输的数据进行加密，而 HASH 算法用于验证数据的完整性。由于浏览器生成的密码是整个数据加密的关键，因此在传输的时候使用非对称加密算法对其加密。非对称加密算法会生成公钥和私钥，公钥只能用于加密数据，可以随意传输，而网站的私钥用于对数据进行解密，所以网站都会非常小心地保管自己的私钥，防止泄漏。

SSL 握手过程中有任何错误都会使加密连接断开，从而阻止隐私信息的传输，正是由于 HTTPS 非常安全，攻击者无法从中找到下手的地方，因此更多地采用假证书的手法来欺骗客户端，从而获取明文的信息。

2.2　请　求　头

请求头描述客户端向服务器发送请求时使用的协议类型、所使用的编码以及发送内容的长度等。客户端（浏览器）通过输入 URL 后确定等于做了一次向服务器的请求动作，在这个请求里面带有请求参数，请求头在网络爬虫中的作用是相当重要的一部分。检测请求头是常见的反爬虫策略，因为服务器会对请求头做一次检测来判断这次请求是人为的还是非人为的。为了形成一个良好的代码编写规范，无论网站是否做 Headers 反爬虫机制，最好每次发送请求都添加请求头。

请求头的参数如下。

（1）Accept：text/html,image/*（浏览器可以接收的文件类型）。

（2）Accept-Charset：ISO-8859-1（浏览器可以接收的编码类型）。

（3）Accept-Encoding：gzip,compress（浏览器可以接收的压缩编码类型）。

（4）Accept-Language：en-us,zh-cn（浏览器可以接收的语言和国家类型）。

（5）Host：请求的主机地址和端口。

（6）If-Modified-Since：Tue, 11 Jul 2000 18:23:51 GMT（某个页面的缓存时间）。

（7）Referer：请求来自于哪个页面的 URL。

（8）User-Agent：Mozilla/4.0（compatible，MSIE 5.5，Windows NT 5.0，浏览器相关信息）。

（9）Cookie：浏览器暂存服务器发送的信息。

（10）Connection：close(1.0)/Keep-Alive(1.1)（HTTP 请求版本的特点）。

（11）Date：Tue, 11 Jul 2000 18:23:51 GMT（请求网站的时间）。

一个标准的请求基本上都带有以上属性。在网络爬虫中，请求头一定要有 User-Agent，其他的属性可以根据实际需求添加，因为反爬虫通常检测请求头的 Referer 和 User-Agent，而 Cookie 不能添加到请求头。除此之外，还有一些比较特殊的请求头信息，如 Upgrade-Insecure-Requests（告诉服务器，浏览器可以处理 HTTPS 协议）、X-Requested-With（判断是否 Ajax 请求）等。

以下是 Python 里面一个完整的请求头，以字典格式生成，代码如下：

```
Headers = {
    'Accept': 'text/html,application/xhtml+xml,
              application/xml;q=0.9 ,*/*;q=0.8',
    'Accept-Language': 'zh-CN,zh;q=0.8',
    'Cache-Control': 'max-age=0',
    'User-Agent': ' Mozilla/5.0 (Windows NT 6.3;
             WOW64; rv:41.0) Gecko/20100101 Firefox/41.0',
    'Connection': 'keep-alive',
       'Referer': 'https://movie.douban.com/'}
```

2.3 Cookies

Cookies 也可以称为 Cookie，指某些网站为了辨别用户身份、进行 Session 跟踪而储存在用户本地终端上的数据。

一个 Cookies 就是存储在用户主机浏览器中的文本文件。Cookies 是纯文本形式，它们不包含任何可执行代码。服务器告诉浏览器将这些信息存储，并且每个请求中都将该信息返回到服务器。服务器之后可以利用这些信息来标识用户。多数需要登录的网站通常会在用户登录后将用户信息写入 Cookies，只要这个 Cookies 存在并且合法，就可以自由地浏览这个网站的所有站点。Cookies 只是包含数据，就其本身而言并不有害。

服务器可以利用 Cookies 包含的信息判断在 HTTP 传输中的状态。Cookies 最典型的应用是判定注册用户是否已经登录网站和保留用户信息以便简化登录手续。

一般 Cookies 所具有的属性如下。

- Domain：域，表示当前 Cookies 属于哪个域或子域下面。
- Path：表示 Cookies 的所属路径。
- Expire Time/Max-Age：表示 Cookies 的有效期。
- Secure：表示该 Cookies 只能用 HTTPS 传输。
- Httponly：表示此 Cookies 必须用 HTTP 或 HTTPS 传输。
- HasKeys：通过该值指示 Cookie 是否含有子键，返回一个 bool 值。
- Name：表示 Cookie 的名称。
- Value：单个 Cookie 的值。
- Values：单个 Cookie 所包含的键值对的集合。

Cookies 的优点如下。

（1）极高的扩展性和可用性。
（2）通过良好地编程控制保存在 Cookie 中的 Session 对象的大小。
（3）通过加密和安全传输技术（SSL）减少 Cookie 被破解的可能性。
（4）只在 Cookie 中存放不敏感数据，即使被盗也不会有重大损失。
（5）可控制 Cookie 的生命期，使之不会永远有效。

Cookies 的缺点如下。

（1）Cookie 数量和长度的限制。每个 domain 最多只能有 20 条 Cookie，每个 Cookie 长度不能超过 4KB，否则会被截掉。

（2）安全性问题。如果 Cookie 被拦截，就有可能被取得所有的 Session 信息。

（3）某些状态不可保存在客户端。例如，为了防止重复提交表单，需要在服务器端保存一个计数器。如果把这个计数器保存在客户端，那么它起不到任何作用。

2.4　HTML

HTML 是超文本标记语言，标准通用标记语言下的一个应用。“超文本”就是指页面内可以包含图片、链接，甚至音乐、程序等非文字元素。超文本标记语言的结构包括“头”部分（Head）和“主体”部分（Body），其中“头”部分提供关于网页的信息，“主体”部分提供网页的具体内容。

爬虫开发对 HTML 的要求是能看懂 HTML 各个标签的含义，了解标签的属性作用以及整个 HTML 布局设计。下面来看一个简单的 HTML 文档的结构：

```
    <!DOCTYPE html> # 声明为 HTML5 文档
    <html># 元素是 HTML 页面的根元素
    <head># 元素包含了文档的元（meta）数据
    <meta charset="utf-8"># 元素可提供有关页面的元信息（meta-information），主要是描述
和关键词
    <title>Python</title># 元素描述了文档的标题
    </head>
    <body> # 元素包含了可见的页面内容
    <h1>我的第一个标题</h1> # 定义一个标题
    <p>我的第一个段落。</p> # 元素定义一个段落
    </body>
    </html>
```

一个完整的网页必定以<html></html>为开头和结尾，整个 HTML 可分为两部分：

（1）<head></head>，主要是对网页的描述、图片和 JavaScript 的引用。<head> 元素包含所有的头部标签元素。在 <head>元素中可以插入脚本（scripts）、样式文件（CSS）及各种 meta 信息。该区域可添加的元素标签有<title>、<style>、<meta>、<link>、<script>、<noscript>和<base>。

（2）<body></body>是网页信息的主要载体。该标签下还可以包含很多类别的标签，不同的标签有不同的作用，标签以<>开头，以</>结尾，<>和</>之间的内容是标签的值和属性，每个标签之间可以是相互独立的，也可以是嵌套、层层递进的关系。

根据这两个组成部分就能很容易地分析整个网页的布局。其中，<body></body>是整个 HTML 的重点部分，通过示例讲述如何分析<body></body>：

```
    <body>
    <h1>我的第一个标题</h1>
    <div>
    <p> Python</p>
    </div>
    <h2>
    <p>
    <a> Python</a>
    </p>
    </h2>
    </body>
```

上述例子分析如下：

（1）<h1>和 是两个不相关的标签，两个标签是相互独立的。

（2） 和<p>是嵌套关系，<p>的上一级标签是 。

（3）<h1>和<p>这两个标签是毫无关系的。

（4）<h2>标签包含一个<p>标签，<p>标签再包含一个<a>标签，一个标签可以包含多个标签在其中。

除上述示例的标签之外，大部分标签都可以在<body></body>中添加，常用的标签如表 2-1 所示。

表 2-1　HTML 常用的标签

HTML 标签	中文释义
Img	图片
A	锚
Strong	加重（文本）
Em	强调（文本）
I	斜体字
B	粗体（文本）
Br	换行
Div	分隔
Span	范围
Ol	排序列表
Ul	不排序列表
Li	列表项目
Dl	定义列表
h1~h6	标题 1 到标题 6
P	段落
Tr	表格中的一行
Th	表格中的表头
Td	表格中的一个单元格

2.5　JavaScript

JavaScript 是一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML 网页上使用的，用来给 HTML 网页增加动态功能。

JavaScript 脚本语言同其他语言一样，有自身的基本数据类型、表达式和算术运算符及程序的基本框架。JavaScript 提供了 4 种基本的数据类型和两种特殊的数据类型用来处理数据和文字。而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。

有时候分析网站需要理解某些 JavaScript 的功能，如某些特殊的数据会存放在 JavaScript 中。以 12306 全国站点为例，如图 2-2 所示。

图 2-2　12306 站点信息

从图 2-2 中可以看到，不同的站点有对应的英文字母，代表站点的编码信息，JavaScript 存储数据主要使用变量的形式。

JavaScript 还能根据用户触发某些事件对用户的操作进行加工处理。例如用户登录信息设置加密处理，原理是先对用户提交的信息做加密处理，然后发送请求到服务器，这一系列事件由 JavaScript 独立完成。要用爬虫实现该功能，就要分析 JavaScript 如何执行整个用户登录过程。

下面通过分析一个简单的例子，来进一步了解 JavaScript 事件的触发原理：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script> <!-- //JavaScript 代码 -→
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == null || x == "") {
        alert("需要输入名字。");
        return false;
    }
    else{alert("你的名字提交成功")}
}
</script>
</head>
<body>
<!-- //html 表单，当点击按钮"提交"后会触发 onsubmit 这个事件，执行 JavaScript 代码 -→
<form name="myForm" action="" onsubmit="return validateForm()" method="post">
名字: <input type="text" name="fname">
<input type="submit" value="提交">
</form>
</body>
</html>
```

JavaScript 事件的触发过程：

（1）HTML 根据<form></form>标签相应地生成一个表单。

（2）当用户在表单输入内容后，单击提交按钮，就会触发<form></form>表单里所指向的 validateForm()方法，执行相应的 JavaScript 代码。

（3）validateForm()会判断输入的值是否为空。如果输入的值为空，就提示输入名字；若输入的值不为空，则提示“提交成功”。

2.6 JSON

JSON（JavaScript Object Notation，JavaScript 对象标记）是一种轻量级的数据交换格式，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言，易于阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。

在 JavaScript 语言中，一切都是对象。因此，任何支持的类型都可以通过 JSON 来表示，例如字符串、数字、对象、数组等。JSON 格式说明如下：

（1）对象表示为键值对。
（2）数据由逗号分隔。
（3）花括号保存对象。
（4）方括号保存数组。

JSON 的书写格式是：键/值对，包括字段名称（字符串），后面写一个冒号，然后是值。例如“name”:“Tom”，等价于 JavaScript 语句：name = “Tom”

JSON 的值可以是数字（整数或浮点数）、字符串、逻辑值（True 或 False）、数组（在方括号中）、对象（在花括号中）和 Null。

例子如下：

```
MyJSon = {
"name": "Python",
"address" :  { "province" :  "广东" , "city" : "广州"}
}
```

JSON 的格式是用花括号表示的，代码 MyJSon 里包含两个属性，分别是 name 和 address。name 的值是“Python”；address 的值是嵌套新的 JSON，里面包含 province 和 city 属性，值为“广东”和“广州”。

一个 JSON 里可以嵌套多个 JSON，也可以嵌套 JSON 数组，都是以键-值的形式表现。在数据结构上，JSON 与 Python 里的字典非常相似。

2.7 Ajax

Ajax 不是一种新的编程语言，而是一种用于创建更好、更快以及交互性更强的 Web 应用程序的技术。使用 JavaScript 向服务器提出请求并处理响应而不阻塞用户，核心对象是 XMLHTTPRequest。通过这个对象，JavaScript 可在不重载页面的情况下与 Web 服务器交换数据，即在不需要刷新页面的情况下就可以产生局部刷新的效果。

Ajax 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可以使网页从服务器请求少量的信息，而不是整个页面。

JavaScript、XML、HTML 与 CSS 在 Ajax 中使用的 Web 标准已被良好定义，并被所有的主流浏览器支持，Ajax 应用程序独立于浏览器和平台。

Web 应用程序比桌面应用程序有优势，能够涉及广大的用户，更易安装及维护，也更易开发。

判断网页数据是否使用 Ajax 最简单的方法是：触发事件之后，判断网页是否发生刷新状态。如果网页没有发生刷新，数据就自动生成，说明数据的加载是通过 Ajax 生成并渲染到网页上的；反之，数据是通过服务器后台生成并加载的。

两种数据加载渲染方式分别由前端和后端完成，实现的方式和原理也不同。判断数据加载方式是爬虫开发必备的基本技能之一，正确地判断数据加载方式才能找到数据来源的渠道，最终才能找到抓取的目标。

2.8　本 章 小 结

本章主要介绍了与编写爬虫程序相关的 Web 前端开发技术。

前端开发技术是爬虫开发人员必备技能之一，也是编写爬虫程序的基础。前端技术的主要作用是分析各类网站的设计架构，以便有针对性地编写爬虫脚本。从整个爬虫开发周期来看，分析网站架构是最为耗时的一环，也是爬虫开发的核心之一，可以说，爬虫的开发都是基于网站的分析为前提。

关于前端开发技术，读者应重点掌握以下内容。

- HTTP 与 HTTPS：互联网上应用最为广泛的一种网络协议。目前所有网站开发都基于该协议，也是网站的实现原理。
- 请求头：基于 HTTP 与 HTTPS 协议实现，其作用是在通信之间实现信息传递。熟知各种请求类型，对爬虫中编写请求头有指导性作用。
- Cookies：存储在用户主机浏览器中的文本文件，主要让服务器识别各个用户身份信息。
- HTML：服务器返回的网页内容，一般由服务器后台生成。网站大部分数据来源于此，熟悉 HTML 布局和各个标签的作用，有利于数据抓取和清洗。
- JavaScript：主要实现网页的动态功能及用户交互。要懂得分析 JavaScript 代码，尤其是数据加密处理。
- JSON：表示一个 JavaScript 对象的信息，本质是一个特殊的字符串。
- Ajax：主要是前端数据加载和渲染技术，其响应内容大部分以 JSON 格式为主。

第 3 章

Chrome 分析网站

3.1　Chrome 开发工具

浏览器是从事编程开发人员必备的开发工具。世界上五大主流浏览器分别是：IE、Opera、Google Chrome、Safari 和 Firefox，其中 Chrome 和 Firefox 是编程开发人员的首选，主要是两者运行速度、扩展性和用户体验都符合开发人员所需。

本书选择 Chrome 作为分析网站的工具，因为其简洁、速度快（无论是启动速度、页面解析速度还是 JavaScript 执行速度），对 HTML5 和 CSS3 的支持也比较完善。

以分析豆瓣电影为例，先打开 Chrome 浏览器，进入豆瓣电影网页（https://movie.douban.com/）。单击 Chrome 的开发者工具（快捷键：F12），如图 3-1 所示。

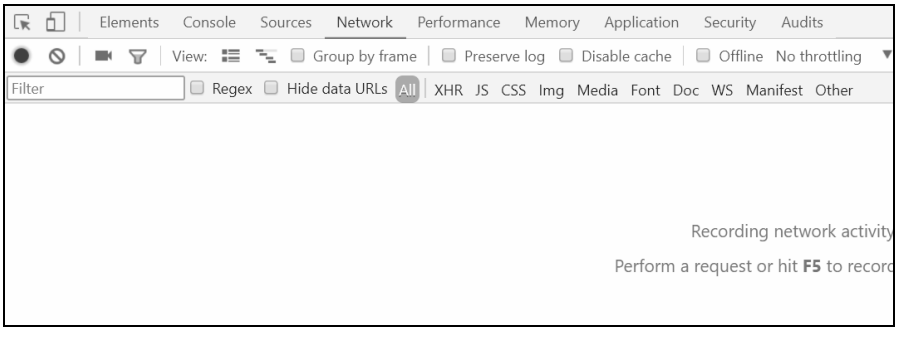


图 3-1　开发者模式

还可以通过在网页上右击，选择“检查”，或者按 Ctrl+Shift+I 组合键，如图 3-2 所示，打开开发者工具界面。

开发者工具的界面共有 9 个标签页，分别是：Elements、Console、Sources、Network、Performance、Memory、Application、Security 和 Audits。

Chrome 开发者工具以 Web 调试为主，如果用于爬虫分析，熟练掌握 Elements 和 Network 标签就能满足大部分的爬虫需求。其中，Network 是核心部分。

图 3-2　开发者模式

3.2　Elements 标签

在 Elements 标签中允许从浏览器的角度看页面，也就是说可以看到 Chrome 渲染页面所需要的 HTML、CSS 和 DOM（Document Object Model）对象。此外，还可以编辑内容更改页面显示效果，如图 3-3 所示。

图 3-3　Elements 标签

图 3-3 中，Elements 标签最左边的按钮用于快速查找网页元素，单击该按钮后，在网页上某一处单击，就会自动显示并选中该元素在 HTML 里的位置。

Elements 标签分成两部分，分别在图 3-3 中标为区域 1 和区域 2，两个区域相辅相成。区域 1 显示整个网页的 HTML 信息，单击选中某一行内容的时候，区域 2 的 Styles 标签会显示当前单击选中内容的 CSS 样式，并可对元素的 CSS 进行查看与编辑修改。Computed 显示当前选中的边距属性、边框属性，用图像显示一个整体效果。Event Listeners 是整个网页事件触发的 JavaScript，如图 3-4 所示。

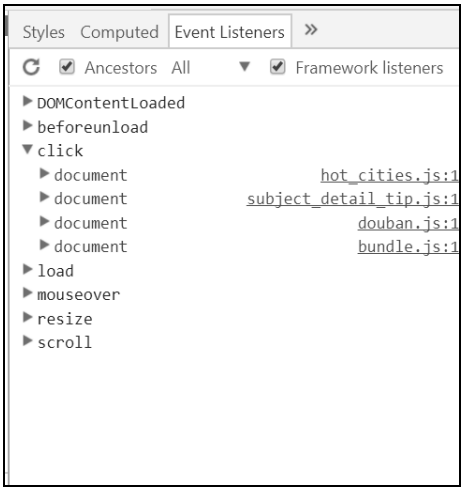


图 3-4　Event Listeners

通过单击 Event Listeners 下的某个 JavaScript 会自动跳转到 Sources 标签，显示当前 JavaScript 的源码，这个功能可快速找到 JavaScript 代码所在的位置，对分析 JavaScript 起到快速定位作用。

3.3　Network 标签

在 Network 标签中可以看到页面向服务器请求的信息、请求的大小以及加载请求花费的时间。从发起网页页面请求 Request 后分析 HTTP 请求得到各个请求信息（包括状态、类型、大小、所用时间、Request 和 Response 等）。Network 标签的结构组成如图 3-5 所示。

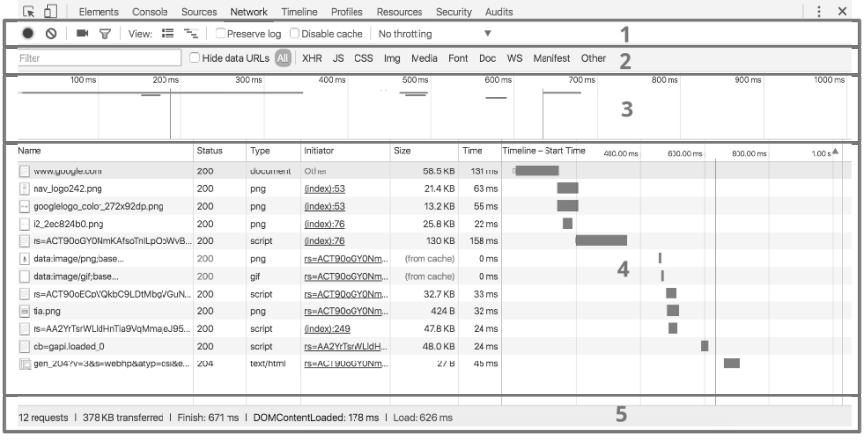


图 3-5　Network 标签

Network 标签主要包括以下 5 个区域。

- Controls：控制 Network 的外观和功能。
- Filters：控制 Requests Table 具体显示哪些内容。

- All：返回当前页面全部加载的信息，就是一个网页全部所需要的代码、图片等请求。
- XHR：筛选 Ajax 的请求链接信息，前面讲过 Ajax 核心对象 XMLHTTPRequest，XHR 取于 XMLHTTPRequest 的缩写。
- JS：主要筛选 JavaScript 文件。
- CSS：主要是 CSS 样式内容。
- Img：是网页加载的图片，爬取图片的 URL 都可以在这里找到。
- Media：是网页加载的媒体文件，如 MP3、RMVB 等音频视频文件资源。
- Doc：是 HTML 文件，主要用于响应当前 URL 的网页内容。

- Overview：显示获取到请求的时间轴信息，主要是对每个请求信息在服务器的响应时间进行记录。这个主要是为网站开发优化方面提供数据参考，这里不做详细介绍。
- Requests Table: 按前后顺序显示所有捕捉的请求信息，单击请求信息可以查看该详细信息。
- Summary：显示总的请求数、数据传输量、加载时间信息。

5 个区域中，Requests Table 是核心部分，主要作用是记录每个请求信息。但每次网站出现刷新时，请求列表都会清空并记录最新的请求信息，如用户登录后发生 304 跳转，就会清空跳转之前的请求信息并捕捉跳转后的请求信息。

对于每条请求信息，可以单击查看该请求的详细信息，如图 3-6 所示。

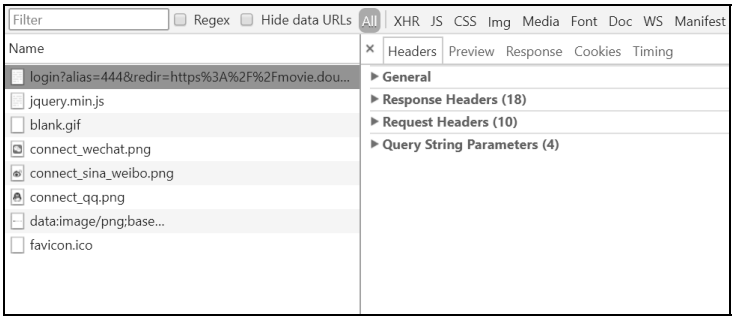


图 3-6　请求信息

每条请求信息划分为以下 5 个标签。

- Headers：该请求的 HTTP 头信息。
- Preview：根据所选择的请求类型（JSON、图片、文本）显示相应的预览。
- Response：显示 HTTP 的 Response 信息。
- Cookies：显示 HTTP 的 Request 和 Response 过程中的 Cookies 信息。
- Timing：显示请求在整个生命周期中各部分花费的时间。

常用的标签有 Headers、Preview 和 Response。Headers 用于获取请求链接、请求头和请求参数；Preview 和 Response 用于显示服务器返回的响应内容。

Headers 标签划分为以下 4 部分。

- General：记录请求链接、请求方式和请求状态码。
- Response Headers：服务器端的响应头，其参数说明如下。
  - Cache-Control：指定缓存机制，优先级大于 Last-Modified。

- Connection：包含很多标签列表，其中最常见的是 Keep-Alive 和 Close，分别用于向服务器请求保持 TCP 连接和断开 TCP 连接。
- Content-Encoding：服务器通过这个头告诉浏览器数据的压缩格式。
- Content-Length：服务器通过这个头告诉浏览器回送数据的长度。
- Content-Type：服务器通过这个头告诉浏览器回送数据的类型。
- Date：当前时间值。
- Keep-Alive：在 Connection 为 Keep-Alive 时，该字段才有用，用来说明服务器估计保留连接的时间和允许后续几个请求复用这个保持着的连接。
- Server：服务器通过这个头告诉浏览器服务器的类型。
- Vary：明确告知缓存服务器按照 Accept-Encoding 字段的内容分别缓存不同的版本。

- Request Headers：用户的请求头。其参数说明如下。
  - Accept：告诉服务器客户端支持的数据类型。
  - Accept-Encoding：告诉服务器客户端支持的数据压缩格式。
  - Accept-Charset：可接受的内容编码 UTF-8。
  - Cache-Control：缓存控制，服务器控制浏览器要不要缓存数据。
  - Connection：处理完这次请求后，是断开连接还是保持连接。
  - Cookie：客户可通过 Cookie 向服务器发送数据，让服务器识别不同的客户端。
  - Host：访问的主机名。
  - Referer：包含一个 URL，用户从该 URL 代表的页面出发访问当前请求的页面，当浏览器向 Web 服务器发送请求的时候，一般会带上 Referer，告诉服务器请求是从哪个页面 URL 过来的，服务器借此可以获得一些信息用于处理。
  - User-Agent：中文名为用户代理，简称 UA，是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等。
- Query String Parameters：请求参数。主要是将参数按照一定的形式（GET 和 POST）传递给服务器，服务器通过接收其参数进行相应的响应，这是客户端和服务端进行数据交互的主要方式之一。

Headers 标签的内容看起来很多，但在实际使用过程中，爬虫开发人员只需关心请求链接、请求方式、请求头和请求参数的内容即可。而 Preview 和 Response 是服务器返回的结果，两者之间对不同类型的响应结果有不同的显示方式：

（1）如果返回的结果是图片，那么 Preview 表示可显示图片内容，Response 表示无法显示。
（2）如果返回的是 HTML 或 JSON，那么两者皆能显示，但在格式上可能会存在细微的差异。

3.4 分析 QQ 音乐

现在以 QQ 音乐某一歌手页面的分析为例（y.qq.com/n/yqq/singer/0025NhlN2yWrP4.html）讲述如何使用 Chrome 开发者工具分析网站，如图 3-7 所示。

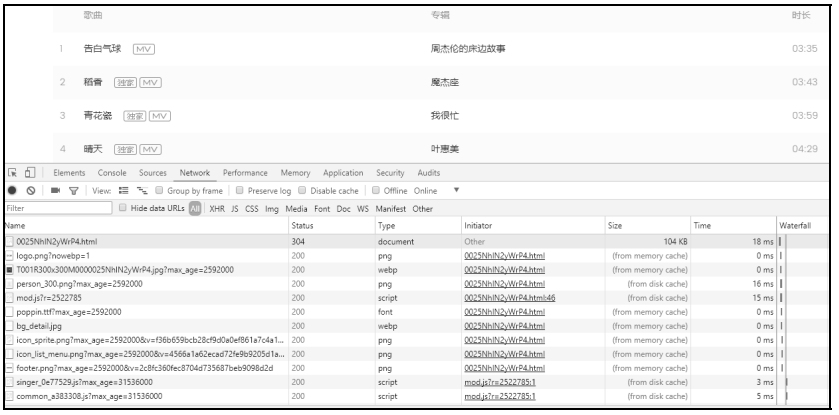


图 3-7　歌手信息

从图 3-7 中可以看到，在 Network 标签下捕捉到很多请求信息，请求类型有 document、png、font 和 script 等，分别对应 HTML 文件、图片、字体格式和 JavaScript 脚本。

单击“Filters”下的 Doc 标签（Doc 是当前网页的 HTML 文件），发现有两个请求信息，分别是“0025NhlN2yWrP4.html”和“xhr_proxy_utf8.html”。从请求的命名可以看出，第一个请求与网站的 URL 是一致的。再查看“0025NhlN2yWrP4.html”的响应内容（Preview 标签），可以使用“Ctrl+F”快速查找歌曲信息，如图 3-8 所示。

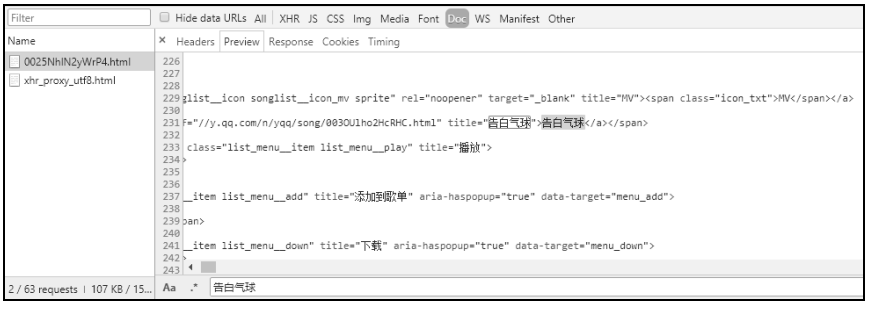


图 3-8　快速查找歌曲信息

在 Doc 中虽能找到歌曲名、专辑和时长，但无法找到更多的歌曲信息。歌曲信息有可能是由其他方式生成的，网站数据生成只有前端（Ajax 或 JSONP）和后端（服务器）两种方式。从图 3-8 返回的结果来看，数据不可能是从后端生成的，那么就可能是由前端加载生成的。

提　示

JSONP（JSON With Padding）是 JSON 的一种“使用模式”，可用于解决主流浏览器的跨域数据访问问题。

前端加载的数据有可能记录在 Chrome 开发者工具的“XHR”或“JS”中，分别查看两个标签里面的请求信息，最终发现歌曲信息存放在 JS 下的某个请求中，如图 3-9 和图 3-10 所示。

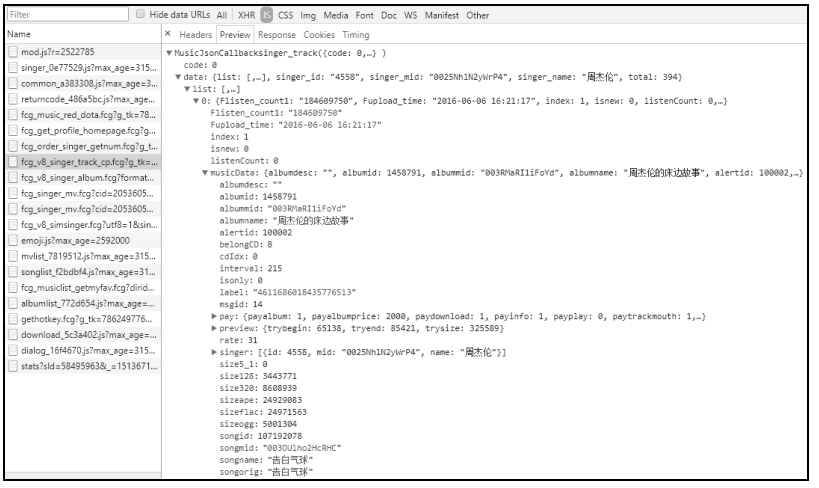


图 3-9　响应内容

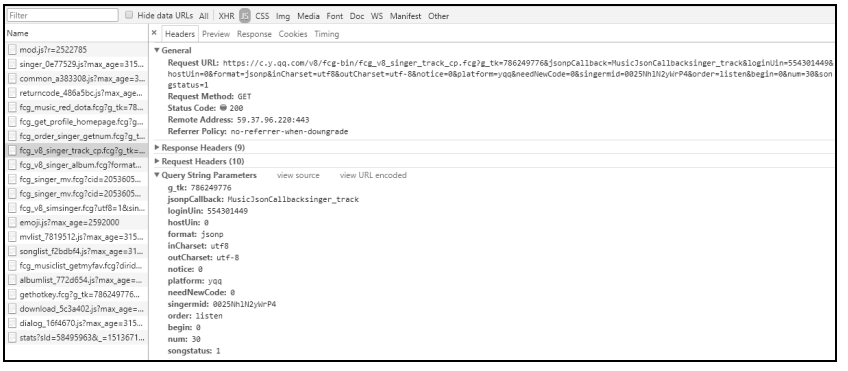


图 3-10　请求信息

从图 3-9 和图 3-10 分析得知，请求方式是 GET，Query String Parameters 是记录该请求的参数。因为请求方式是 GET，所以请求参数也可以在请求链接上找到。

再看请求参数，大部分请求参数是可以明确知道的，唯独参数 singermid 无法确定。从参数的命名来看，这应该是歌手的 ID 信息，也是网站用于标记歌手唯一的属性，所以要获取歌手的 singermid 可能需要从其他请求上获取。

根据上述例子，可简单总结出分析网站的步骤如下：

步骤01 找出数据来源，大部分数据来源于 Doc、XHR 和 JS 标签。

步骤02 找到数据所在的请求，分析其请求链接、请求方式和请求参数。

步骤03 查找并确定请求参数来源。有时候某些请求参数是通过另外的请求生成的，比如请求 A 的参数 id 是通过请求 B 所生成的，那么要获取请求 A 的数据，就要先获取请求 B 的数据作为 A 的请求参数。

3.5　本 章 小 结

Chrome 开发者工具的主要作用是进行 Web 开发调试，对于爬虫开发人员来说，应该熟练掌握 Elements、Console 和 Network。其中 Network 是核心部分，百分之九十的网站分析都在 Network 上完成，读者对 Network 上的各个功能和作用要理解掌握，并懂得如何使用 Chrome 分析网站的请求信息。

一般分析网站最主要的是找到数据的来源，确定数据来源就能确定数据生成的具体方法。总结归纳分析网站的步骤如下：

（1）找出数据来源，大部分数据来源于 Doc、XHR 和 JS 标签。

（2）找到数据所在的请求，分析其请求链接、请求方式和请求参数。

（3）查找并确定请求参数来源。有时候某些请求参数是通过另外的请求生成的，比如请求 A 的参数 id 是通过请求 B 所生成的，那么要获取请求 A 的数据，就要先获取请求 B 的数据作为 A 的请求参数。

上述分析步骤适用于大部分网站，但每个网站都有自身的设计特点，不能一概而论。此方法更多的是起到指导性作用，遇到具体的问题还是要具体分析。

第 4 章

Fiddler 抓包

4.1 Fiddler 介绍

Fiddler 是一款非常流行并且实用的 HTTP 抓包工具，原理是在电脑上开启一个 HTTP 代理服务器，然后转发所有的 HTTP 请求和响应。因此，比一般的浏览器自带的抓包工具（开发者工具）要好用得多。不仅如此，还可以支持请求重放一些高级功能，也可以支持对手机应用进行 HTTP 抓包。

Fiddler 是用 C#开发的工具，包含一个简单却功能强大的基于 JScript .NET 事件的脚本子系统，灵活性非常棒，可以支持众多的 HTTP 调试任务，并且能够使用.net 框架语言进行扩展。

此外，还支持断点调试技术，当请求或响应属性能够跟目标的标准相匹配时，Fiddler 就能够暂停 HTTP 通信，并且允许修改请求和响应。这种功能对于安全测试非常有用，当然也可以用来做一般的功能测试。

4.2 Fiddler 安装配置

Fiddler 在 Windows 下可直接使用 exe 安装包安装，安装包可在官方网站下载（https://www.telerik.com/download/fiddler）。

完成安装后，在安装目录下双击打开应用程序 Fiddler.exe，可看到 Fiddler 用户界面，如图 4-1 所示。

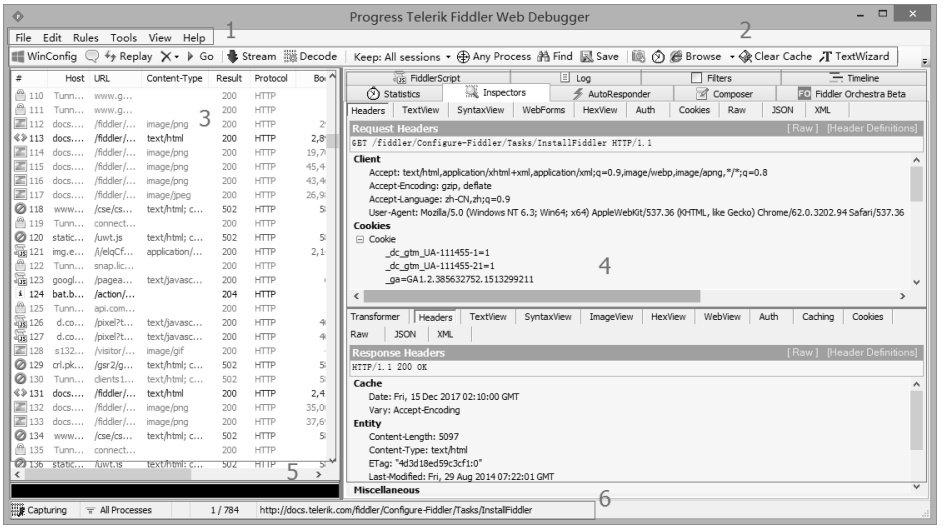


图 4-1　Fiddler 用户界面

Fiddler 用户界面主要包括下面 6 个部分：

（1）图中标注 1 为 Main Menu（主菜单），作用于整个 Fiddler 相关配置。

（2）图中标注 2 为 Toolbar（工具栏），主要对 Web Session 操作处理。

（3）图中标注 3 为 Web Session（列表），显示已抓取的 HTTP 请求信息。

（4）图中标注 4 为 View（选项视图），显示每条 HTTP 的详细信息。

（5）图中标注 5 为 Quickexec（命令行），通过特定的条件快速找到符合条件的 HTTP 请求。

（6）图中标注 6 为 Status bar（状态栏），显示当前状态信息。

打开 Fiddler 之后，由于 HTTPS 协议的特殊性，还需要配置 Fiddler。了解 Fiddler 抓取 HTTPS 协议的原理才能更好地理解如何对 Fiddler 进行配置，原理如图 4-2 所示。

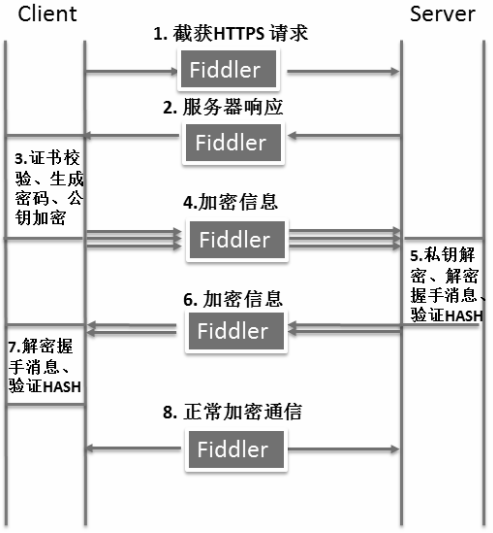


图 4-2　Fiddler 抓取 HTTPS 的原理

Fiddler 抓取 HTTPS 协议充当的角色：

（1）服务器→客户端：Fiddler 接收到服务器发送的密文，用对称密钥解开，获得服务器发送的明文。再次加密，发送给客户端。

（2）客户端→服务器端：客户端用对称密钥加密，被 Fiddler 截获后，解密获得明文。再次加密，发送给服务器端。由于 Fiddler 一直拥有通信用对称密钥 enc_key，因此在整个 HTTPS 通信过程中信息对其透明。

配置 Fiddler，使其能够抓取 HTTPS 请求信息，方法如下：

步骤01 对 Fiddler 进行设置：打开 Main Menu→Tools→Fiddler Options→HTTPS。

步骤02 勾选 HTTPS 里的选项，然后单击 Actions→Trust Root Certificate，完成证书验证，如图 4-3 所示。

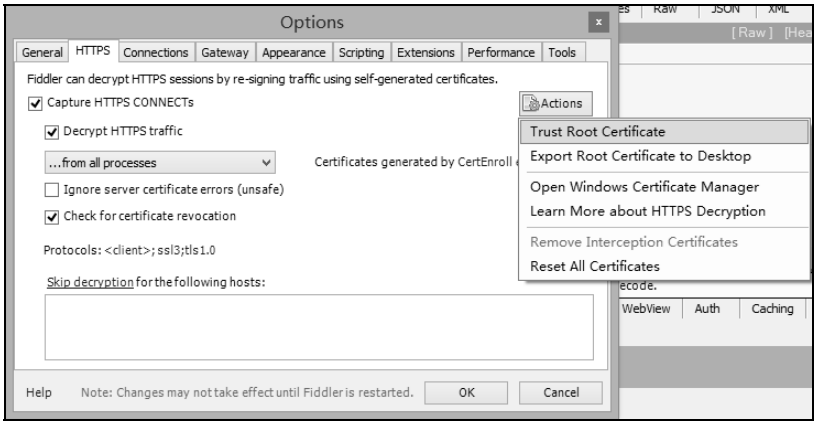


图 4-3　Fiddler 配置 HTTPS

步骤03 完成配置，重启浏览器，Fiddler 就能正常抓取 HTTPS 请求信息。

完成上述安装和配置，Fiddler 就能抓取浏览器的请求信息。除此之外，Fiddler 还能抓取手机上的请求信息，具体使用方法在后续章节会详细讲述。

4.3　Fiddler 抓取手机应用

Fiddler 可通过同一无线网络实现对手机应用的抓包，手机抓包原理和电脑抓包原理相同，手机抓包主要通过远程连接实现手机和 Fiddler 通信。

实现 Fiddler 抓取手机应用的步骤如下：

步骤01 配置 Fiddler 远程连接模式。打开 Main Menu→Tools→Fiddler Options→Connections，勾选 Allow remote computers to connect 复选框，如图 4-4 所示。

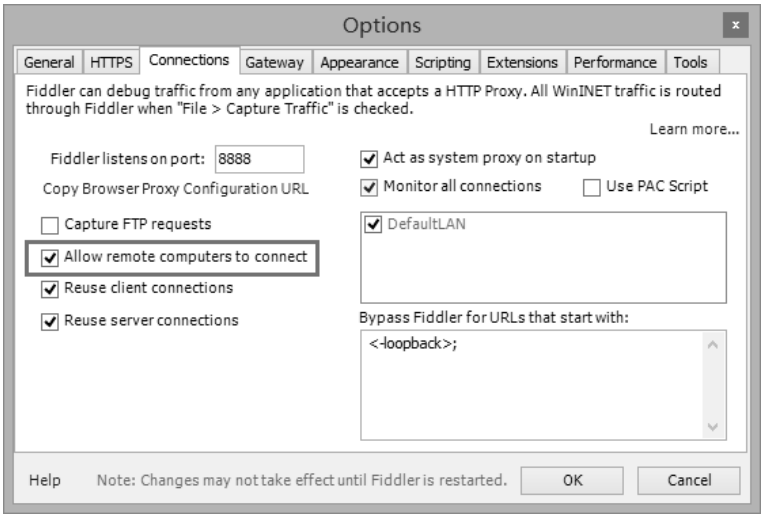


图 4-4　Fiddler 配置远程连接

步骤02 在手机端进行参数配置（以安卓手机为例）。确保手机和电脑在同一个网络，查询电脑 IP 地址，可在 CMD 下输入 ipconfig 查询（电脑 IP 为 10.168.1.240），从图 4-4 得知，Fiddler 端口为 8888（一般默认为 8888，也可自行设置）。

步骤03 在手机浏览器中输入电脑 IP 地址和 Fiddler 端口（输入“10.168.1.240: 8888”），单击确认后跳转到证书下载页面。单击下载 FiddlerRoot certificate，如图 4-5 所示。

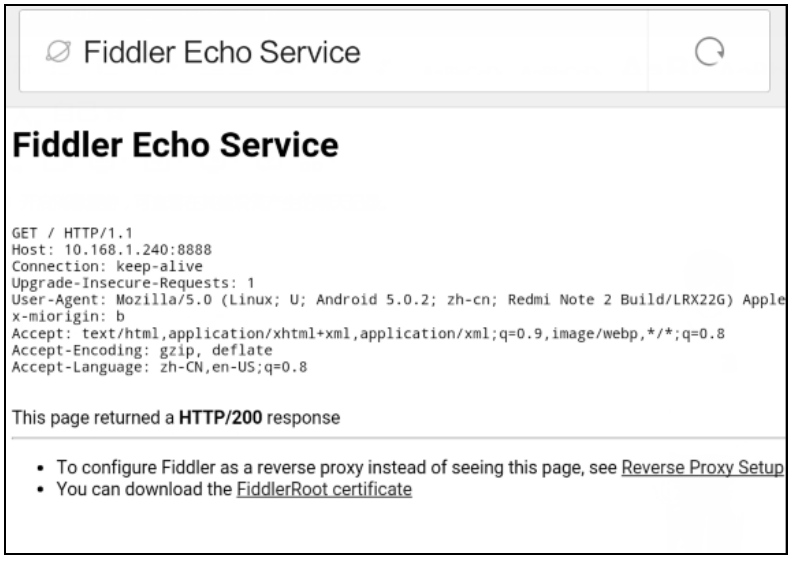


图 4-5　下载 FiddlerRoot certificate

步骤04 证书文件以 cer 为后缀名，由于不同的手机型号安装证书的方式不一致，因此这里不做详细讲述。完成证书安装后，进入手机当前连接 Wi-Fi 详情，设置代理 IP：主机名为电脑 IP 地址，端口为 Fiddler 配置的端口，如图 4-6 所示。

图 4-6　配置手机代理

步骤05 完成上述配置，可操作手机应用，在操作过程中产生的 HTTP 请求都会被电脑上的 Fiddler 抓取，如图 4-7 所示。

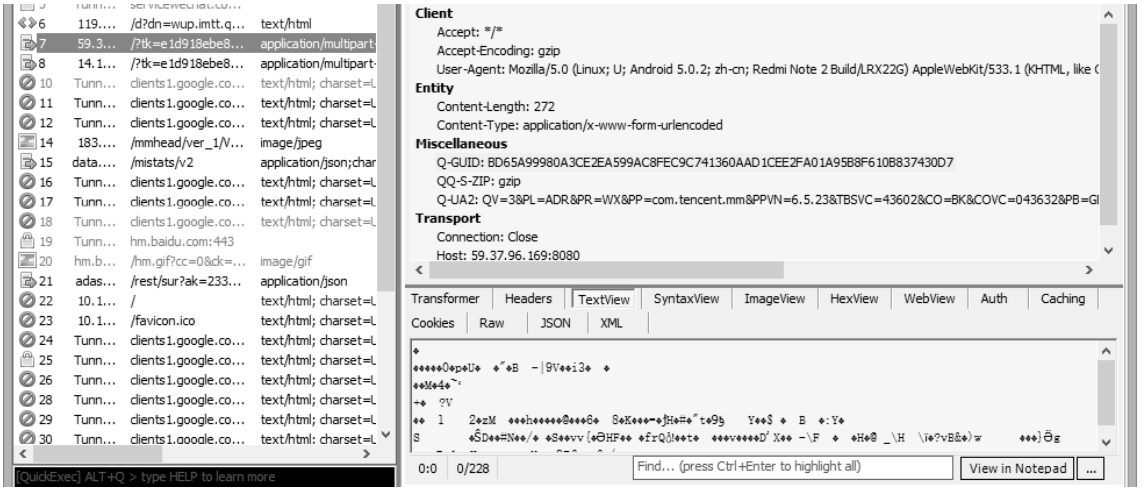


图 4-7　Fiddler 抓取手机 HTTP 请求信息

从图 4-7 中看到，User-Agent 请求头是由安卓系统发出的。同样地，若抓取 iOS 系统，也是按照上述方式配置即可。

如果停止电脑对手机的网络监控，可以回到步骤 4，将 Wi-Fi 的代理设置去掉即可。若要删除 Fiddler 证书，可在设置→系统安全→信任的凭据下找到“DO_NOT_TRUST”证书，将其删除即可。

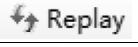
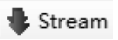
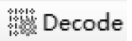
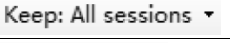
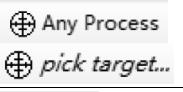
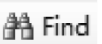
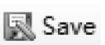
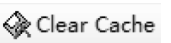
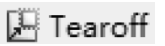
4.4　Toolbar 工具栏

现在已可以使用 Fiddler 抓取 HTTP 请求信息了，但如何使用 Fiddler 对请求进行分析，从而获取我们所需的信息呢？要熟练掌握 Fiddler 的使用，首先要掌握其每个功能的作用。

从 4.2 节知道，Fiddler 用户界面由 6 部分组成，最为常用的是 Toolbar 工具栏、Web Session 列表、View 选项视图和 Quickexec 命令行。

Toolbar 工具栏主要提供常见的命令和设置的快捷方式，其各个按钮的功能说明如表 4-1 所示。

表 4-1　Toolbar 工具栏功能说明

快捷键	含义
	单击该按钮可以为所有选定的 Session 添加 comment
	向服务器重新发送该请求
	从 Web Session 中删除已经捕捉的 Session
	恢复执行在 request 或 response 断点处暂停的所有 Session
	打开 Stream 模式，取消所有没有设置中断的缓存
	打开 Decode 模式，对请求和响应的 HTTP 内容和传输编码进行解码
	选择 Web Session 列表中保存 Session 的数量
	单击上面的 Any Process 图标并将其移动到指定浏览器页面后，该 log 会单独记录这个页面的通信情况
	打开 Find Session 窗口，可快速查找某条 Session
	把所有的 Session 保存到 saz 文件中
	把当前桌面的屏幕截图以 JPEG 格式添加到 Web Session 列表
	简单的计时功能
	如果选中某个 Session，就会在 IE 中打开该 URL；如果没有选中 Session，就在 IE 中打开 about:blank
	清空 WinINET 的缓存文件
	打开文本编码/解码小工具
	新建一个包含所有 View 的窗口
	在 MSDN 的 Web Session 区域进行搜索
	打开 Fiddler 的帮助窗口
	删除工具栏，如果要恢复工具栏，可单击 View→Show Toolbar

4.5 Web Session 列表

Web Session 主要以表格形式展现，需掌握表字段代表的含义、表格信息的含义以及快捷键的使用。

表字段（表头）信息的含义如表 4-2 所示。

表 4-2 Web Session 表头信息及说明

表头	含义与作用
#	对已捕捉的 Session 生成对应的 ID 号
Host	接受请求的主机名和端口
URL	请求 URL 的路径
Content-Type	Session 的内容类型
Result	响应的状态码
Protocol	网络协议类型（HTTP、HTTPS、FTP）
Body	包含的字节数
Caching	响应头中 Expires 和 Cache-Control 的值
Process	数据流在本地系统的进程
Comments	通过工具栏 Comment 按钮设置注释信息
Custom	FiddlerScript 所设置的 Ui-CustomColumn 标志位的值

观察列表每条请求信息可发现，每条数据都有不同的颜色，其颜色含义如表 4-3 所示。

表 4-3 Web Session 请求信息的颜色含义

颜色	含义
红色	表示 HTTP 状态错误
黄色	表示 HTTP 状态需用户认证
灰色	表示数据流类型 CONNECT 或表示响应内容是图像
紫色	表示响应内容是 CSS 文件
蓝色	表示响应内容是 HTML
绿色	表示响应内容是 Script 文件

除了颜色之外，每条请求信息都带有一个图标，图标含义如表 4-4 所示 。

表 4-4 Web Session 请求信息的图标含义

图标	含义
	正在向服务器发送请求
	正在向服务器获取响应
	请求停止于断点处，允许对它进行修改
	响应停止于断点处，允许对它进行修改

（续表）

图标	含义
	请求使用的是 Head 或 Options 方法，客户端无须下载内容即可获取目标 URL 和服务器信息
	请求使用 POST 方法向服务器发送数据
	响应内容是 HTML
	响应内容是图像文件
	响应内容是脚本文件
	响应内容是 CSS 文件
	响应内容是 XML 格式文件
	响应内容是 JSON 格式文件
	响应内容是音频文件
	响应内容是视频文件
	响应内容是 SilverLight 程序
	响应内容是 Flash 应用程序
	响应内容是字体文件
	响应成功
	Content-Type 内容是 Text/HTML
	响应是 HTTP / 300、301、302、303、307 重定向
	要求对客户端进行认证
	服务器返回错误的标识
	Session 被客户端或服务器中止
	响应内容使用缓存版本

通过对表字段、请求信息的颜色和请求信息的图标的了解，可知道 Fiddler 对每一种请求类型都进行了详细的划分。除此之外，用户还可以对每个请求进行操作，移动鼠标对某一条请求右击会出现操作菜单，如图 4-8 所示。

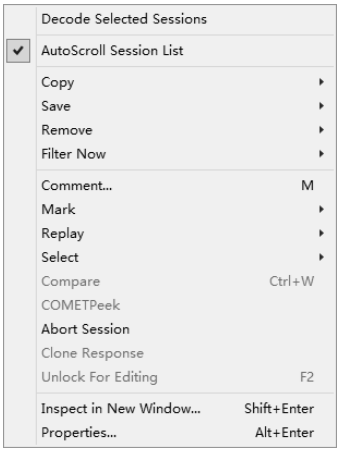


图 4-8　Web Session 操作菜单

用户可通过操作菜单对请求信息进行更改，也可以直接使用快捷键实现，快捷键如表 4-5 所示。（图中颜色较深的是常用部分。）

表 4-5 Web Session 快捷键

快捷键	含义
SpaceBar	在视图中激活并显示当前的 Session
Ctrl + A	选中所有的 Session
ESC	取消选择所有的 Session
Ctrl + I	反向选中，取消选中的 Session，选中之前未选中的 Session
Ctrl + X	删除所有 Session
Delete	删除选中的 Session
Shift + Delete	删除所有未选中的 Session
R	重新执行当前请求
Shift + R	多次执行当前的请求（在提示框输入执行次数）
U	无条件重新执行当前的请求
Shift +U	无条件多次执行当前的请求（在提示框输入执行次数）
P	选中触发该请求的父请求
C	选中该响应触发的所有子请求
D	选中与当前 Session 重复的请求
Alt + Enter	查看当前 Session 的属性
Shift + Enter	在新的 Fiddler 窗口中启动该 Session 的 Inspectors
Ctrl + 1/2/3/4/5/6	选中的 Session 分别用粗体的红色/蓝色/金色/绿色/橙色/紫色表示
M	为选中的 Session 添加描述

4.6 View 选项视图

View 主要以选项视图方式实现，将一个请求信息按功能划分在不同选项卡中，如表 4-6 所示是常用的选项视图。

表 4-6 View 常用选项视图

常用选项视图	含义
Statistics	统计选项卡，统计资源的消耗时间和数据长度等信息
Inspectors	检查选项卡，共分为两部分：请求信息和响应信息
AutoResponder	自动响应选项卡，将请求重定向到本地文件，实现人工干预 HTTP 请求
Composer	构建选项卡，用于创建 HTTP Request，并发送服务器实现模拟请求

使用 Fiddler 做爬虫开发必须掌握 Inspectors、AutoResponder 和 Composer。Inspectors 主要是对请求和响应信息进行分析和获取请求参数，如图 4-9 所示。

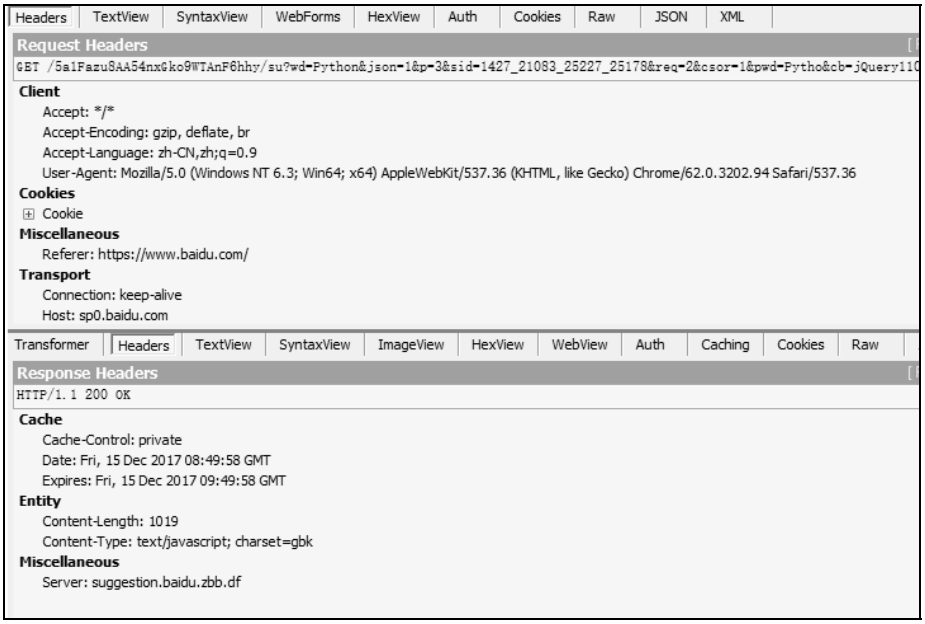


图 4-9　Inspectors 选项视图

从图 4-9 知道，Inspectors 上下划分为两个功能区。上面的功能区显示用户发送的请求信息，下面的功能区显示服务器响应的内容，每个功能区里又划分了多个选项视图。其中，Headers 选项视图显示请求头和响应头，WebForm 选项视图显示发送请求的请求参数，xxxView 选项卡显示各种类型的数据内容。

AutoResponder 和 Composer 就不多做讲解了，主要是 AutoResponder 的作用更偏向于 Web 开发调试，在爬虫开发中实用性不强；Composer 虽然能够实现对服务器的请求，但用 Fiddler 编写代码就显得本末倒置，还不如直接使用 Python 实现。

4.7　Quickexec 命令行

Quickexec 命令行可通过特定的条件快速找到符合条件的请求信息，如图 4-10 所示。

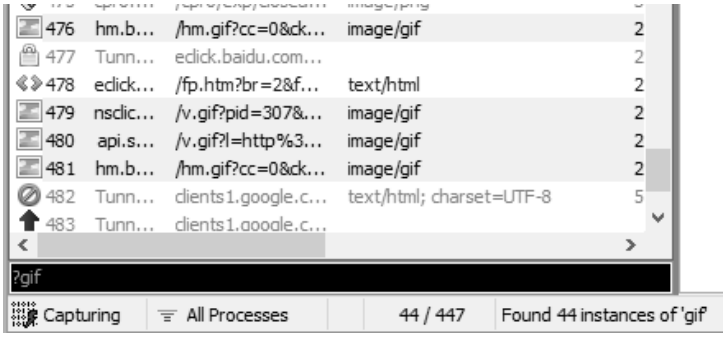


图 4-10　Quickexec 快速查找

从图 4-10 中看到，当输入“?gif”时，Web Session 列表会将符合条件的请求信息以高亮显示，在下方状态栏可以看到符合条件的请求有 44 条。

除了使用 Quickexec 实现快速查找之外，还可以使用“Ctrl+F”查找功能，如图 4-11 所示。

通过输入关键字，然后单击查找按钮，就能找到相对应的 Session 信息，而且还可以自行设定目标高亮颜色。除此之外，使用者还能根据特定的要求设置不同的查找条件。

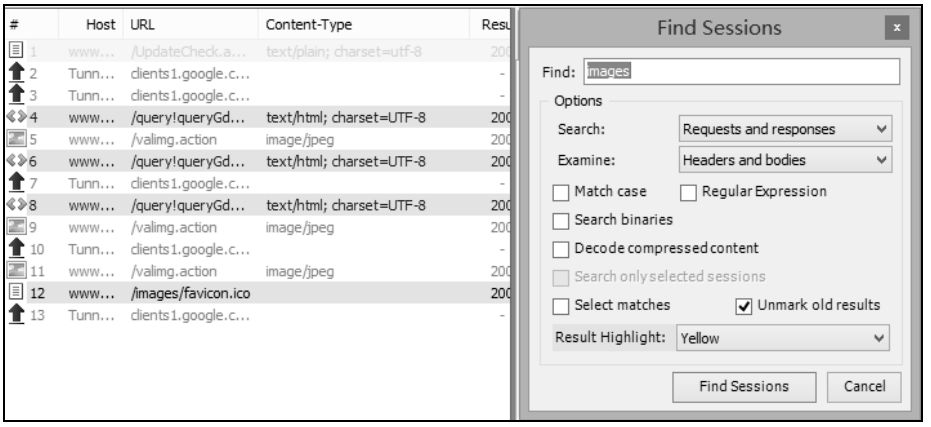


图 4-11　Ctrl+F 查找功能

4.8　本 章 小 结

Fiddler 是一款非常流行并且实用的 HTTP 抓包工具，它的原理是在电脑中开启一个 HTTP 代理服务器，然后转发所有的 HTTP 请求和响应。因此，比一般的浏览器自带的抓包工具（开发者工具）要好用得多。不仅如此，Fiddler 还可以支持请求重放等一些高级功能，也可以支持对手机应用进行 HTTP 抓包。

Fiddler 提供了 Windows 环境下的.exe 安装包，使其安装极其简单方便。安装完成后，需配置 HTTPS 抓取功能和手机抓包功能，完成配置便可对 HTTPS 网站和手机进行抓包。

除了功能强大之外，在使用上也较为简单，使用者只要打开 Fiddler，然后在浏览器（手机）中进行操作，Fiddler 就会自动抓取请求信息。就 Fiddler 本身的功能而言，使用者只需熟知每个功能按钮的作用便知道如何使用。

对于爬虫开发人员来说，需要掌握 Web Session 列表和 View 常用选项视图的基本功能，能够分析并得知每个请求的类型、状态码、请求方式、请求头、请求链接、请求参数以及响应内容等基本信息。

第5章

爬虫库 Urllib

5.1 Urllib 简介

Urllib 是 Python 自带的标准库，无须安装，直接引用即可。Urllib 通常用于爬虫开发、API（应用程序编程接口）数据获取和测试。在 Python 2 和 Python 3 中，Urllib 在不同版本中的语法有明显的改变。

Python 2 分为 Urllib 和 Urllib2，Urllib2 可以接收一个 Request 对象，并以此来设置一个 URL 的 Headers，但是 Urllib 只接收一个 URL，意味着不能伪装用户代理字符串等。Urllib 模块可以提供进行 Urlencode 的方法，该方法用于 GET 查询字符串的生成，Urllib2 不具有这样的功能。这也是 Urllib 与 Urllib2 经常在一起使用的原因。

在 Python 3 中，Urllib 模块是一堆可以处理 URL 的组件集合，就是将 Urllib 和 Urllib2 合并在一起使用，并且命名为 Urllib。

由于 Urllib 在不同的 Python 版本上有明显的区别，在实际开发中也遇到一些尴尬的情况，其中最为主要的是版本之间的互不兼容所带来的问题。

在 Python 3 中，Urllib 是一个收集几个模块来使用 URL 的软件包，大致具备以下功能。

- urllib.request：用于打开和读取 URL。
- urllib.error：包含提出的例外 urllib.request。
- urllib.parse：用于解析 URL。
- urllib.robotparser：用于解析 robots.txt 文件。

5.2 发 送 请 求

urllib.request.urlopen 的语法如下：

```
    urllib.request.urlopen(url, data=None, [timeout, ]*, cafile=None, capath=N
one, cadefault=False, context=None)
```

功能说明：Urllib 是用于访问 URL（请求链接）的唯一方法。

【参数解释】

- url：需要访问的网站的 URL 地址。url 格式必须完整，如 https://movie.douban.com/为完整的 url，若 url 为 movie.douban.com/，则程序运行时会提示无法识别 url 的错误。
- data：默认值为 None，Urllib 判断参数 data 是否为 None 从而区分请求方式。若参数 data 为 None，则代表请求方式为 GET；反之请求方式为 POST，发送 POST 请求。参数 data 以字典形式存储数据，并将参数 data 由字典类型转换成字节类型才能完成 POST 请求。
- timeout：超时设置，指定阻塞操作（请求时间）的超时（如果未指定，就使用全局默认超时设置）。
- cafile、capath 和 cadefault：使用参数指定一组 HTTPS 请求的可信 CA 证书。cafile 应指向包含一组 CA 证书的单个文件；capath 应指向证书文件的目录；cadefault 通常使用默认值即可。
- context：描述各种 SSL 选项的实例。

在实际使用中，常用的参数有 url、data 和 timeout。若在爬虫中遇到证书验证，则可将证书验证直接关闭，也可以设置参数指向证书的信息和位置。相比而言，设置证书比较耗时，而且通用性不强。

当对网站发送请求时，网站会返回相应的响应内容。urlopen 对象提供获取网站响应内容的方法函数，分别介绍如下。

- read()、readline()、readlines()、fileno()和 close()：对 HTTPResponse 类型数据操作。
- info()：返回 HTTPMessage 对象，表示远程服务器返回的头信息。
- getcode()：返回 HTTP 状态码。
- geturl()：返回请求的 URL。

下面的例子用于实现 Urllib 模块对网站发送请求并将响应内容写入文本文档，代码如下：

```
    # 导入 urllib
    import urllib.request
    # 打开 URL
    response = urllib.request.urlopen('https://movie.douban.com/', None, 2)
    # 读取返回的内容
    html = response.read().decode('utf8')
    # 写入 txt
    f = open('html.txt', 'w', encoding='utf8')
    f.write(html)
    f.close()
```

首先导入 urllib.request 模块，然后通过 urlopen 访问一个 URL，请求方式是 GET，所以参数 data 设置为 None；最后的参数用于设置超时时间，设置为 2 秒，如果超过 2 秒，网站还没返回响应数据，就会提示请求失败的错误信息。

当得到服务器的响应后，通过 response.read()获取其响应内容。read()方法返回的是一个 bytes 类型的数据，需要通过 decode()来转换成 str 类型。最后将数据写入文本文档中，encoding 用于设置文本文档的编码格式，数据编码必须与文本文档编码一致，否则会出现乱码。运行结果如图 5-1 所示。

图 5-1　获取豆瓣页面内容

5.3　复杂的请求

urllib.request.Request 的语法如下：

```
urllib.request.Request(url, data=None, headers={}, method=None)
```

功能说明：声明一个 request 对象，该对象可自定义 header（请求头）等请求信息。

【参数解释】

- url：完整的 url 格式，与 urllib.request.urlopen 的参数 url 一致。
- data：请求参数，与 urllib.request.urlopen 的参数 data 一致。
- headers：设置 request 请求头信息。
- method：设定请求方式，主要是 POST 和 GET 方式。

一个完整的 HTTP 请求必须要有请求头信息，而 urllib.request.Request 的作用是设置 HTTP 的请求头信息。使用 urllib.request.Request 为 5.2 节的例子设置请求头，代码如下：

```
# 导入 urllib
import urllib.request
url = 'https://movie.douban.com/'
# 自定义请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64)
                  AppleWebKit/537.36 (KHTML, like Gecko)
                  Chrome/45.0.2454.85 Safari/537.36
```

```
                    115Browser/6.0.3',
    'Referer': 'https://movie.douban.com/',
    'Connection': 'keep-alive'}
# 设置 request 的请求头
req = urllib.request.Request(url, headers=headers)
# 使用 urlopen 打开 req
html = urllib.request.urlopen(req).read().decode('utf-8')
# 写入文件
f = open('html.txt', 'w', encoding='utf8')
f.write(html)
f.close()
```

5.4 代理 IP

代理 IP 的原理：以本机先访问代理 IP，再通过代理 IP 地址访问互联网，这样网站（服务器）接收到的访问 IP 就是代理 IP 地址。

Urllib 提供了 urllib.request.ProxyHandler()方法可动态设置代理 IP 池，代理 IP 主要以字典格式写入方法。完成代理 IP 设置后，将设置好的代理 IP 写入 urllib.request.build_opener()方法，生成对象 opener，然后通过 opener 的 open()方法向网站（服务器）发送请求。

沿用前面章节的例子，将例子改为使用代理 IP 访问网站，代码如下：

```
import urllib.request
url = 'https://movie.douban.com/'
# 设置代理 IP
proxy_handler = urllib.request.ProxyHandler({
    'http': '218.56.132.157:8080',
    'https': '183.30.197.29:9797'})
# 必须使用 build_opener()函数来创建带有代理 IP 功能的 opener 对象
opener = urllib.request.build_opener(proxy_handler)
response = opener.open(url)
html = response.read().decode('utf-8')
f = open('html.txt', 'w', encoding='utf8')
f.write(html)
f.close()
```

注意，由于使用代理 IP，因此连接 IP 的时候有可能出现超时而导致报错，遇到这种情况只要更换其他代理 IP 地址或者再次访问即可。以下是常见的报错信息。

- ConnectionResetError: [WinError 10054]远程主机强迫关闭了一个现有的连接。
- urllib.error.URLError: urlopen error Remote end closed connection without response（结束没有响应的远程连接）。
- urllib.error.URLError: urlopen error [WinError 10054] 远程主机强迫关闭了一个现有的连接。
- TimeoutError: [WinError 10060] 由于连接方在一段时间后没有正确答复或连接的主机没有反应，因此连接尝试失败。
- urllib.error.URLError: urlopen error [WinError 10061] 由于目标计算机拒绝访问，因此无法连接。

5.5　使用 Cookies

Cookies 主要用于获取用户登录信息，比如，通过提交数据实现用户登录之后，会生成带有登录状态的 Cookies，这时可以将 Cookies 保存在本地文件中，下次程序运行的时候，可以直接读取 Cookies 文件来实现用户登录。特别对于一些复杂的登录，如验证码、手机短信验证登录这类网站，使用 Cookies 能简单解决重复登录的问题。

Urllib 提供 HTTPCookieProcessor()对 Cookies 操作，但 Cookies 的读写是由 MozillaCookieJar()完成的。下面的例子实现 Cookies 写入文件，代码如下：

```
import urllib.request
from http import cookiejar
filename = 'cookie.txt'
# MozillaCookieJar 保存 cookie
cookie = cookiejar.MozillaCookieJar(filename)
# HTTPCookieProcessor 创建 cookie 处理器
handler = urllib.request.HTTPCookieProcessor(cookie)
# 创建自定义 opener
opener = urllib.request.build_opener(handler)
# open 方法打开网页
response = opener.open('https://movie.douban.com/')
# 保存 cookie 文件
cookie.save()
```

代码中的 cookiejar 是自动处理 HTTP Cookie 的类，MozillaCookieJar()用于将 Cookies 内容写入文件。程序运行时先创建 MozillaCookieJar()对象，然后将对象直接传入函数 HTTPCookieProcessor()，生成 opener 对象；最后使用 opener 对象访问 URL，访问过程所生成的 Cookies 就直接写入已创建的文本文档中。

接着再看如何读取 Cookies，代码如下：

```
import urllib.request
from http import cookiejar
filename = 'cookie.txt'
# 创建 MozillaCookieJar 对象
cookie = cookiejar.MozillaCookieJar()
# 读取 cookie 内容到变量
cookie.load(filename)
# HTTPCookieProcessor 创建 cookie 处理器
handler = urllib.request.HTTPCookieProcessor(cookie)
# 创建 opener
opener = urllib.request.build_opener(handler)
# opener 打开网页
response = opener.open('https://movie.douban.com/')
# 输出结果
print(cookie)
```

读取和写入的方法很相似，主要区别在于：两者对 MozillaCookieJar()对象的操作不同，导致实现功能也不同。运行结果如图 5-2 所示。

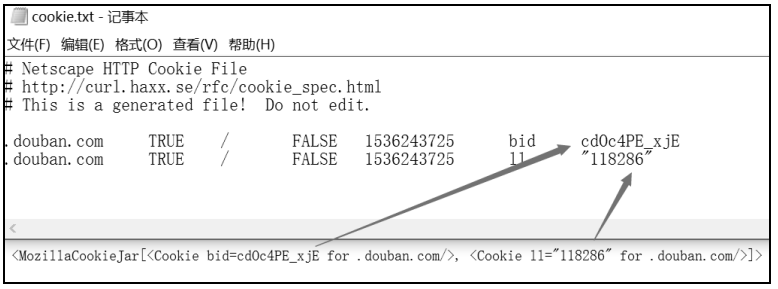


图 5-2 验证 Cookies

注意，为了方便测试，上述代码中使用的 cookie.save()和 cookie.load(filename)将 Cookies 内容显示在文本文档中。在实际开发中，为了提高安全性，可以在保存和读取 Cookies 时设置参数，使 Cookies 信息隐藏在文件中。方法如下：

```
cookie.save(ignore_discard=True,ignore_expires=True)
cookie.load(filename, ignore_discard=True, ignore_expires=True)
```

5.6 证书验证

当遇到一些特殊的网站时，在浏览器上会显示连接不是私密连接而导致无法浏览该网页。若在没有安装 12306 根证书的情况下访问 12306 网站，则页面如图 5-3 所示。

图 5-3 查询 12306 的车票

这里补充一个知识点，CA 证书也叫 SSL 证书，是数字证书的一种，类似于驾驶证、护照和营业执照的电子副本。因为配置在服务器上，也称为 SSL 服务器证书。

SSL 证书就是遵守 SSL 协议，由受信任的数字证书机构颁发 CA，在验证服务器身份后颁发，具有服务器身份验证和数据传输加密功能。

SSL 证书在客户端浏览器和 Web 服务器之间建立一条 SSL 安全通道（Secure Socket Layer，SSL），安全协议是由 Netscape Communication 公司设计开发的。该安全协议主要用来提供对用户和服务器的认证，对传送的数据进行加密和隐藏，确保数据在传送中不被改变，即数据的完整性，现已成为该领域中全球化的标准。

一些特殊的网站会使用自己的证书，如 12306 首页提示下载安装根证书，这是为了确保网站

的数据在传输过程中的安全性。在讲述 urllib.request.urlopen 的时候，urlopen 带有 cafile、capath 和 cadefault 参数，可以用于设置用户的 CA 证书。

遇到这类验证证书的网站，最简单而暴力的方法是直接关闭证书验证，可以在代码中引入 SSL 模块，设置关闭证书验证即可。代码如下：

```
import urllib.request
import ssl
# 关闭证书验证
ssl._create_default_https_context=ssl._create_unverified_context
url = 'https://kyfw.12306.cn/otn/leftTicket/init'
response = urllib.request.urlopen(url)
# 输出状态码
print(response.getcode())
```

5.7　数 据 处 理

我们知道 urllib.request.urlopen()方法是不区分请求方式的，识别请求方式主要通过参数 data 是否为 None。如果向服务器发送 POST 请求，那么参数 data 需要使用 urllib.parse 对参数内容进行处理。

Urllib 在请求访问服务器的时候，如果发生数据传递，就需要对内容进行编码处理，将包含 str 或 bytes 对象的两个元素元组序列转换为百分比编码的 ASCII 文本字符串。如果字符串要用作 POST，那么它应该被编码为字节，否则会导致 TypeError 错误。

Urllib 发送 POST 请求的方法如下：

```
import urllib.request
import urllib.parse
url = 'https://movie.douban.com/'
data = {
    'value': 'true',
}
# 数据处理
data = urllib.parse.urlencode(data).encode('utf-8')
req = urllib.request.urlopen(url, data=data)
```

代码中 urllib.parse.urlencode(data)将数据转换成字节的数据类型，而 encode('utf-8')设置字节的编码格式。这里需要注意的是，编码格式主要根据网站的编码格式来决定。urlencode()的作用只是对请求参数做数据格式转换处理。

除此之外，Urllib 还提供 quote()和 unquote()对 URL 编码处理，使用方法如下：

```
import urllib.parse
url = '%2523%25E7%25BC%2596%25E7%25A8%258B%2523'
# 第一次解码
first = urllib.parse.unquote(url)
print(first)
# 输出：'%23%E7%BC%96%E7%A8%8B%23'
# 第二次解码
second = urllib.parse.unquote(first)
```

```
print(second)
# 输出：'#编程#'
```

上述例子将已编码处理的 URL 进行解码还原，同样的方法，可使用 quote()对数据进行编码处理。quote()和 unquote()的作用是解决请求参数中含有中文内容的问题。

5.8 本 章 小 结

本章主要讲解了 Python 自带模块 Urllib 的功能和使用。Urllib 通常用于爬虫开发和 API（应用程序编程接口）数据获取和测试。在 Python 2 和 Python 3 中，Urllib 的语法有明显的改变。其常用的语法有以下几种。

- urllib.request.urlopen：urllib 最基本的使用功能，用于访问 URL（请求链接）的唯一方法。
- urllib.request.Request：声明 request 对象，该对象可自定义请求头（header）、请求方式等信息。
- urllib.request.ProxyHandler：动态设置代理 IP 池，可加载请求对象。
- urllib.request.HTTPCookieProcessor：设置 Cookies 对象，可加载请求对象。
- urllib.request.build_opener()：创建请求对象，用于代理 IP 和 Cookies 对象加载。
- urllib.parse.urlencode(data).encode('utf-8')：请求数据格式转换。
- urllib.parse.quote(url)：URL 编码处理，主要对 URL 上的中文等特殊符号编码处理。
- urllib.parse.unquote(url)：URL 解码处理，将 URL 上的特殊符号还原。

除了 Urllib 之外，一些特殊请求需要结合其他模块配合使用，如 Cookies 读写由 HTTP 模块完成，关闭证书验证需要 SSL 模块设置，等等。