# 课程介绍

* 1. 爬虫课程的特点：

爬虫经常需要爬取商业网站或政府网站的内容，而这些网站随时可能进行更新，网络原因和网站反爬虫机制也会对爬虫代码演示造成干扰。每次上课之前会先测试备课代码，但是如果网站更新发生在授课期间或者反爬虫机制触发，我们尽量在短时间内尝试解决。如果问题很棘手，那么课堂上，请不要过度纠结这个具体问题。

* 网络爬虫介绍（了解）
* HTTP和HTTPS基础知识（掌握）
* Fiddler软件安装与使用（了解）
* urllib标准库使用（掌握）
* 部分常见反爬虫机制以及应对方法（掌握）

1. 网络爬虫介绍
   1. 概念

网络爬虫：按照一定的规则，自动地抓取互联网信息的程序或者脚本。

* 1. 为什么要学爬虫
     + 1. 有不少职位涉及爬虫工程，不一定非得是专职爬虫工程师，有更多的职位会兼职编写复杂度相对较低的爬虫，比如商业数据分析师等职位。
       2. 还有不少生活福利，比如12306抢票、比如各种优惠各种薅羊毛、比如自动爬取豆瓣害羞组福利等等。
  2. 用途

我们经常听说移动互联网已经瓜熟蒂落了、移动互联网红利已然开发殆尽，移动互联网时代即将结束，未来是“大数据时代”或者人工智能时代，至于是会在大数据时代停留很久，还是直接跑步进入人工智能时代，没人敢确定。但可以确定的是，人工智能也需要海量的数据。那么，我们所需要的海量数据从何而来？

网络爬虫可以提供一种非常重要的来源。

* 1. 分类
     1. 通用爬虫

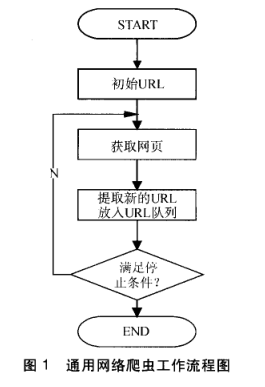
通用网络爬虫是捜索引擎抓取系统（Baidu、Google、Sogou等）的一个重要组成部分。主要目的是将互联网上的网页下载到本地，形成一个互联网内容的镜像备份。然后就可以将这些网页信息用于为搜索引擎建立索引从而提供支持，它决定着整个引擎系统的内容是否丰富，信息是否即时，因此其性能的优劣直接影响着搜索引擎的效果。

* + - 1. **通用搜索引擎（Search Engine）工作原理**

**第一步：抓取网页**

搜索引擎网络爬虫的基本工作流程如下：

1. 首先选取一部分的种子URL，将这些URL放入待抓取URL队列；
2. 取出待抓取URL，解析DNS得到主机的IP，并将URL对应的网页下载下来，存储进已下载网页库中，并且将这些URL放进已抓取URL队列。
3. 分析已抓取URL队列中的URL，分析其中的其他URL，并且将URL放入待抓取URL队列，从而进入下一个循环....



搜索引擎如何获取一个新网站的URL：

1. 新网站向搜索引擎主动提交网址：（如百度<http://zhanzhang.baidu.com/linksubmit/url>）

2. 在其他网站上设置新网站外链（尽可能处于搜索引擎爬虫爬取范围）

3. 搜索引擎和DNS解析服务商(如DNSPod等）合作，新网站域名将被迅速抓取。

但是搜索引擎蜘蛛的爬行是被输入了一定的规则的，它需要遵从一些命令或文件的内容，如标注为nofollow的链接，或者是Robots协议。

Robots协议（也叫爬虫协议、机器人协议等），全称是“网络爬虫排除标准”（Robots Exclusion Protocol），网站通过Robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取，例如：

淘宝网：<https://www.taobao.com/robots.txt>

腾讯网： <http://www.qq.com/robots.txt>

**第二步：数据存储**

搜索引擎通过爬虫爬取到的网页，将数据存入原始页面数据库。其中的页面数据与用户浏览器得到的HTML是完全一样的。

搜索引擎蜘蛛在抓取页面时，也做一定的重复内容检测，一旦遇到访问权重很低的网站上有大量抄袭、采集或者复制的内容，很可能就不再爬行。

**第三步：预处理**

搜索引擎将爬虫抓取回来的页面，进行各种步骤的预处理。

* 中文分词
* 消除噪音（比如版权声明、导航条、广告等……）
* 索引处理
* 链接关系计算
* 特殊文件处理
* ....

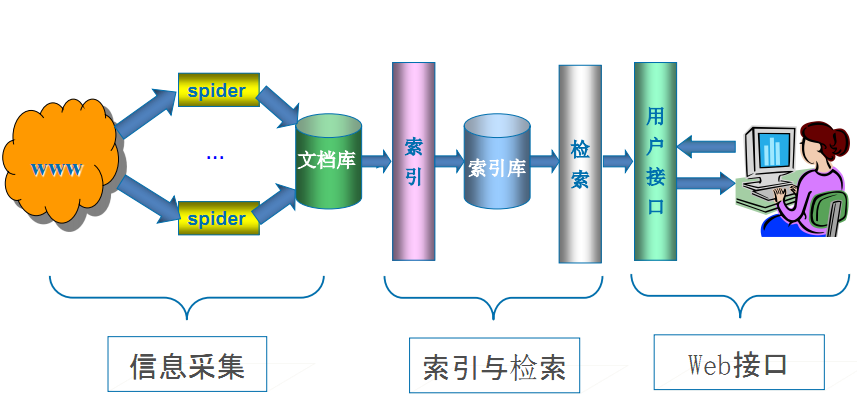
除了HTML文件外，搜索引擎通常还能抓取和索引以文字为基础的多种文件类型，如 PDF、Word、WPS、XLS、PPT、TXT 文件等。我们在搜索结果中也经常会看到这些文件类型。

但搜索引擎还不能处理图片、视频、Flash 这类非文字内容，也不能执行脚本和程序。

**第四步：提供检索服务，网站排名**

搜索引擎在对信息进行组织和处理后，为用户提供关键字检索服务，将用户检索相关的信息展示给用户。

同时会根据页面的PageRank值（链接的访问量排名）来进行网站排名，这样Rank值高的网站在搜索结果中会排名较前，当然也可以直接使用 Money 购买搜索引擎网站排名，简单粗暴。



但是，这些通用性搜索引擎也存在着一定的局限性：

1. 通用搜索引擎所返回的结果都是网页，而大多情况下，网页里90%的内容对用户来说都是无用的。
2. 不同领域、不同背景的用户往往具有不同的检索目的和需求，搜索引擎无法提供针对具体某个用户的搜索结果。
3. 万维网数据形式的丰富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，通用搜索引擎对这些文件无能为力，不能很好地发现和获取。
4. 通用搜索引擎大多提供基于关键字的检索，难以支持根据语义信息提出的查询，无法准确理解用户的具体需求。
5. 聚焦爬虫

聚焦爬虫，是"面向特定需求"的一种网络爬虫程序，它与通用搜索引擎爬虫的区别在于： 聚焦爬虫在实施网页抓取时会对内容进行处理筛选，尽量保证只抓取与需求相关的网页信息。因此他克服了通用爬虫的局限性，更有针对性。

我们课程后续学习的，是聚焦爬虫。

* 1. URL（了解）
     1. 概念

URL（Uniform / Universal Resource Locator的缩写）：统一资源定位符，是用于完整地描述Internet上网页和其他资源的地址的一种标识方法。

举例

<https://movie.douban.com/review/9533560/>

<https://market.douban.com/?utm_campaign=douban_top_nav&utm_source=douban&utm_medium=pc_web>

* + 1. 一般格式（带方括号[]的为可选项）

scheme:// host [:port] / path / [;parameters][?query][#anchor]

* + 1. 组成
       1. scheme：协议，例如 http, https, ftp, file, ed2k, …
       2. host：域名或IP地址（有时需要包含端口号）
       3. port：服务器的端口（如果是走协议默认端口，缺省端口80）
       4. path：访问资源的路径
       5. query-string：参数，发送给http服务器的数据
       6. anchor：锚点（跳转到网页的指定锚点位置）
    2. URL实例

例如：

ftp://219.219.216.11:8080/incoming

https://www.douban.com

http://item.jd.com/12186192.html#product-detail

* 1. 回顾HTTP和HTTPS
     1. HTTP的概念

HTTP协议（HyperText Transfer Protocol，超文本传输协议）：是一种发布和接收HTML页面的方法。

* + 1. 客户端HTTP请求

URL只是标识资源的位置，而HTTP是用来提交和获取资源。客户端发送一个HTTP请求到服务器的请求消息，包括以下格式：

请求行、请求头部、空行、请求数据

四个部分组成，下图给出了请求报文的一般格式。



**一个典型的HTTP请求示例**

**请求方法**

GET https://www.baidu.com/ HTTP/1.1

Host: www.baidu.com

Connection: keep-alive

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8

Referer: http://www.baidu.com/

Accept-Encoding: gzip, deflate, sdch, br

Accept-Language: zh-CN,zh;q=0.8,en;q=0.6

Cookie: BAIDUID=04E4001F34EA74AD4601512DD3C41A7B:FG=1; BIDUPSID=04E4001F34EA74AD4601512DD3C41A7B; PSTM=1470329258; MCITY=-343%3A340%3A; BDUSS=nF0MVFiMTVLcUh-Q2MxQ0M3STZGQUZ4N2hBa1FFRkIzUDI3QlBCZjg5cFdOd1pZQVFBQUFBJCQAAAAAAAAAAAEAAADpLvgG0KGyvLrcyfrG-AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFaq3ldWqt5XN; H\_PS\_PSSID=1447\_18240\_21105\_21386\_21454\_21409\_21554; BD\_UPN=12314753; sug=3; sugstore=0; ORIGIN=0; bdime=0; H\_PS\_645EC=7e2ad3QHl181NSPbFbd7PRUCE1LlufzxrcFmwYin0E6b%2BW8bbTMKHZbDP0g; BDSVRTM=0

GET https://www.baidu.com/ HTTP/1.1

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP 0.9：只有基本的文本 GET 功能。

HTTP 1.0：完善的请求/响应模型，并将协议补充完整，定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP 1.1：在 1.0 基础上进行更新，新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

HTTP 2.0（未普及）：请求/响应首部的定义基本没有改变，只是所有首部键必须全部小写，而且请求行要独立为 :method、:scheme、:host、:path这些键值对。

| **序号** | **方法** | **描述** |
| --- | --- | --- |
| 1 | GET | 请求指定的页面信息，并返回实体主体。 |
| 2 | HEAD | 类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头 |
| 3 | POST | 向指定资源提交数据进行处理请求（例如提交表单或者上传文件），数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。 |
| 4 | PUT | 从客户端向服务器传送的数据取代指定的文档的内容。 |
| 5 | DELETE | 请求服务器删除指定的页面。 |
| 6 | CONNECT | HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。 |
| 7 | OPTIONS | 允许客户端查看服务器的性能。 |
| 8 | TRACE | 回显服务器收到的请求，主要用于测试或诊断。 |

HTTP请求主要分为Get和Post两种方法

* GET是从服务器上获取数据，POST是向服务器传送数据
* GET请求参数显示，都显示在浏览器网址上，HTTP服务器根据该请求所包含URL中的参数来产生响应内容，即“Get”请求的参数是URL的一部分。 例如： http://www.baidu.com/s?wd=Chinese
* POST请求参数在请求体当中，消息长度没有限制而且以隐式的方式进行发送，通常用来向HTTP服务器提交量比较大的数据（比如请求中包含许多参数或者文件上传操作等），请求的参数包含在“Content-Type”消息头里，指明该消息体的媒体类型和编码，

**注意：避免使用Get方式提交表单，因为有可能会导致安全问题。 比如说在登陆表单中用Get方式，用户输入的用户名和密码将在地址栏中暴露无遗。**

* + 1. 常用的请求报头 （考虑加标题）

**1. Host (主机和端口号)**

Host：对应网址URL中的Web名称和端口号，用于指定被请求资源的Internet主机和端口号，通常属于URL的一部分。

**2. Connection (链接类型)**

Connection：表示客户端与服务连接类型

1. Client 发起一个包含 Connection:keep-alive 的请求，HTTP/1.1使用 keep-alive 为默认值。
2. Server收到请求后：
   * 如果 Server 支持 keep-alive，回复一个包含 Connection:keep-alive 的响应，不关闭连接；
   * 如果 Server 不支持 keep-alive，回复一个包含 Connection:close 的响应，关闭连接。
3. 如果client收到包含 Connection:keep-alive 的响应，向同一个连接发送下一个请求，直到一方主动关闭连接。

keep-alive在很多情况下能够重用连接，减少资源消耗，缩短响应时间，比如当浏览器需要多个文件时(比如一个HTML文件和相关的图形文件)，不需要每次都去请求建立连接。

3. Upgrade-Insecure-Requests (升级为HTTPS请求)

Upgrade-Insecure-Requests：升级不安全的请求，意思是会在加载 http 资源时自动替换成 https 请求，让浏览器不再显示https页面中的http请求警报。

*HTTPS 是以安全为目标的 HTTP 通道，所以在 HTTPS 承载的页面上不允许出现 HTTP 请求，一旦出现就是提示或报错。*

**4. User-Agent (浏览器名称)**

User-Agent：是客户浏览器的名称，以后会详细讲。

5. Accept (传输文件类型)

Accept：指浏览器或其他客户端可以接受的MIME（Multipurpose Internet Mail Extensions（多用途互联网邮件扩展））文件类型，服务器可以根据它判断并返回适当的文件格式。

举例：

Accept: \*/\*：表示什么都可以接收。

Accept：image/gif：表明客户端希望接受GIF图像格式的资源；

Accept：text/html：表明客户端希望接受html文本。

Accept: text/html, application/xhtml+xml;q=0.9, image/\*;q=0.8：表示浏览器支持的 MIME 类型分别是 html文本、xhtml和xml文档、所有的图像格式资源。

q是权重系数，范围 0 =< q <= 1，q 值越大，请求越倾向于获得其“;”之前的类型表示的内容。若没有指定q值，则默认为1，按从左到右排序顺序；若被赋值为0，则用于表示浏览器不接受此内容类型。

Text：用于标准化地表示的文本信息，文本消息可以是多种字符集和或者多种格式的；Application：用于传输应用程序数据或者二进制数据。

6. Referer (页面跳转处)

Referer：表明产生请求的网页来自于哪个URL，用户是从该 Referer页面访问到当前请求的页面。这个属性可以用来跟踪Web请求来自哪个页面，是从什么网站来的等。该字段经常用于防盗用。

在下载某些网站的图片时，可能会遇到需要提供referer字段的情况，否则无法下载，那是因为人家做了防盗链，原理就是根据referer去判断是否是本网站的地址，如果不是，则拒绝，如果是，就可以下载；

**7. Accept-Encoding（文件编解码格式）**

Accept-Encoding：指出浏览器可以接受的编码方式。编码方式不同于文件格式，它是为了压缩文件并加速文件传递速度。浏览器在接收到Web响应之后先解码，然后再检查文件格式，许多情形下这可以减少大量的下载时间。

举例：Accept-Encoding:gzip;q=1.0, identity; q=0.5, \*;q=0

如果有多个Encoding同时匹配, 按照q值顺序排列，本例中按顺序支持 gzip, identity压缩编码，支持gzip的浏览器会返回经过gzip编码的HTML页面。 如果请求消息中没有设置这个域服务器假定客户端对各种内容编码都可以接受。

8. Accept-Language（语言种类）

Accept-Langeuage：指出浏览器可以接受的语言种类，如en或en-us指英语，zh或者zh-cn指中文，当服务器能够提供一种以上的语言版本时要用到。

9. Accept-Charset（字符编码）

Accept-Charset：指出浏览器可以接受的字符编码。

举例：Accept-Charset:iso-8859-1,gb2312,utf-8

* ISO8859-1：通常叫做Latin-1。Latin-1包括了书写所有西方欧洲语言不可缺少的附加字符，英文浏览器的默认值是ISO-8859-1.
* gb2312：标准简体中文字符集;
* utf-8：UNICODE 的一种变长字符编码，可以解决多种语言文本显示问题，从而实现应用国际化和本地化。

如果在请求消息中没有设置这个域，缺省是任何字符集都可以接受。

**10. Cookie （Cookie）**

Cookie：浏览器用这个属性向服务器发送Cookie。Cookie是在浏览器中寄存的小型数据体，它可以记载和服务器相关的用户信息，也可以用来实现会话功能，以后会详细讲。

11. Content-Type (POST数据类型)

Content-Type：POST请求里用来表示的内容类型。

举例：Content-Type = Text/XML; charset=gb2312：

指明该请求的消息体中包含的是纯文本的XML类型的数据，字符编码采用“gb2312”。

* + 1. 服务端HHHdHTTP响应

HTTP响应也由四个部分组成，分别是： 状态行、消息报头、空行、响应正文**：**

状态行和请求行对应，响应报头和请求报头类似的结构，此处就不详述了。

**响应状态码**

响应状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值。

**常见状态码：**

* 100~199：表示服务器成功接收部分请求，要求客户端继续提交其余请求才能完成整个处理过程。
* 200~299：表示服务器成功接收请求并已完成整个处理过程。常用200（OK 请求成功）。
* 300~399：为完成请求，客户需进一步细化请求。例如：请求的资源已经移动一个新地址、常用302（所请求的页面已经临时转移至新的url）、307和304（使用缓存资源）。
* 400~499：客户端的请求有错误，常用404（服务器无法找到被请求的页面）、403（服务器拒绝访问，权限不够）。
* 500~599：服务器端出现错误，常用500（请求未完成。服务器遇到不可预知的情况）。
  + 1. HHHdCookie 和 Session：

服务器和客户端的交互仅限于请求/响应过程，结束之后便断开，在下一次请求时，服务器会认为新的客户端。

为了维持服务器和客户端之间的链接，让服务器知道这是刚才同一个用户发送的请求，必须在某处保存客户端的信息。

**Cookie**：客户端记录的信息，用于证明用户的身份。

**Session**：服务器端记录的信息，用于验证用户的身份。

* 1. urllib包
     1. 概念

urllib包（package），python3的标准库，由多个与URL处理相关的模块（module）组成：

* + - 1. **urllib.request**

用来获取URL对应的资源对象。

* + - 1. urllib.error

包含了urllib.request可能抛出的异常。

* + - 1. urllib.parse

用来解析URL。

* + - 1. urllib.robotparser

用来解析robots.txt文件。robots.txt文件是网站的爬虫协议文件。

* + - 1. urllib.response

urllib内部使用的Response类，用于提供对返回的响应提供最基础处理方法的接口，完全附属于urllib.requests模块使用。

当前主要使用了urllib.request模块；以后会逐渐用到其它模块。

* + 1. 发展

Python3中的urllib包整合了Python2中原来的urllib和urllib2等模块。

这部分内容在Python2和Python3之间变化很大，甚至Python3的某些小版本之间也有明显变化，遇到问题尽量查阅对应版本的官方文档，而不是baidu，因为baidu得到的文章很可能是使用Python2的，会让自己更糊涂。同时，这也正是一个锻炼自己阅读官方文档的好机会。

* + 1. urllib.request模块中的重点组件
       1. urlopen函数

urllib.request.urlopen(url, …)

* + - 1. Request类

urllib.request.Request

* 1. urllib.request.urlopen函数

这是urllib.request模块中的最重要函数。

* + 1. 作用

向指定的url发送访问请求，返回该url对应体（资源、服务器等）对该访问请求后产生的访问结果。

* + 1. 使用方法

**urllib.request.urlopen(url, data, timeout, …)**

url是唯一必给的参数，可以是url地址的字符串，也可以是Request对象。

data是传送数据，可选参数。

timeout是设置超时等待时间，可选。

其他参数请自行参阅官方文档。

返回一个响应对象，对于http和https请求来说，返回http.client.HTTPResponse对象，用得最多的是他的.read()方法。

* + 1. http.client.HTTPResponse类

对于http和https请求，urlopen()函数的返回值是一个http.client.HTTPResponse类的实例。http.client.HTTPResponse有如下方法或属性最常用到：

* + - 1. .read()：读入并返回响应体（response body）
      2. .getheader(name): 返回name对应的响应头字段
      3. .status：状态码

常用的就这几个，别的请参考官方文档：

https://docs.python.org/3.6/library/http.client.html

* + 1. 案例 直接使用url地址获取豆瓣Top250电影列表页面



* 1. urllib.request.Request类

urllib.request中最重要的类。

* + 1. 对象的创建

**Request(url, data, headers, …)**

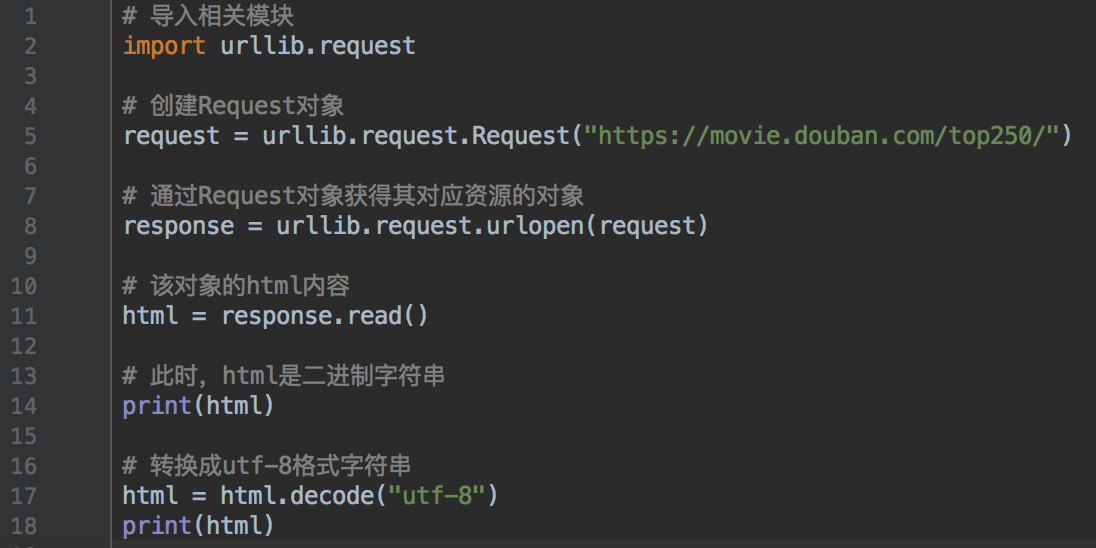
url是必需指定的参数，为url地址的字符串。

data可选参数，默认空，如果传入非空值，HTTP请求将从"GET"方式改为"POST"方式。

Headers可选参数，默认空，需要传入一个字典，包含了需要发送的HTTP报头的键值对。

其他参数请自行参阅官方文档。

* + 1. 案例 使用Request对象获取豆瓣Top250电影列表页面



* 1. User-Agent

这是HTTP的访问报头（request headers，可以在用urllib.request.Request类创建对象时通过headers参数指定）之一。

* + 1. 作用

宣称自己是什么类型的访问者，如某个版本的浏览器，等等。

这是爬虫对抗反爬虫技术的第一大招。

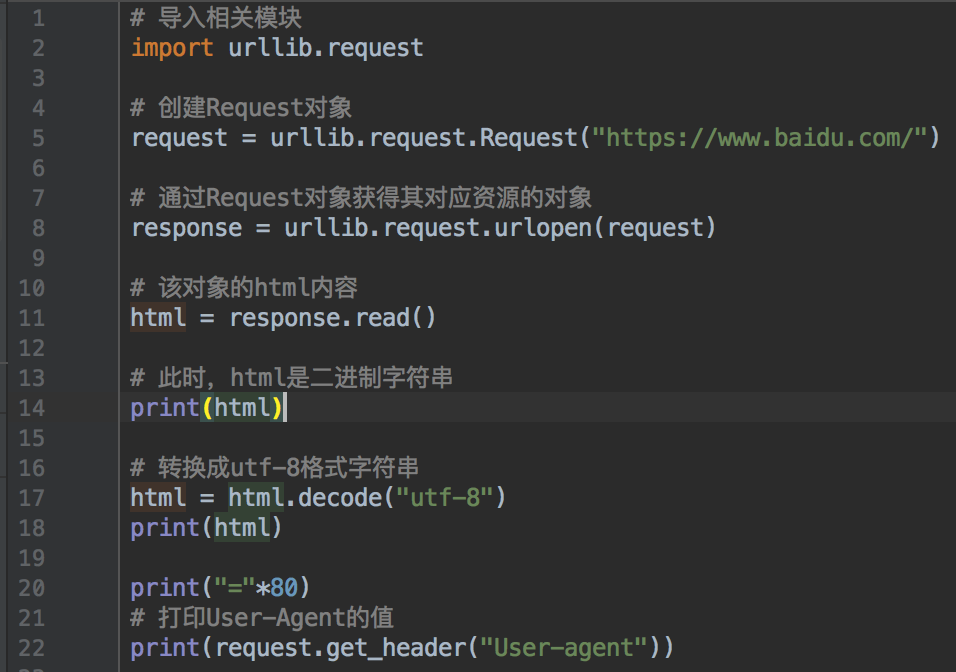
* + 1. 案例 直接使用url地址获取百度首页页面

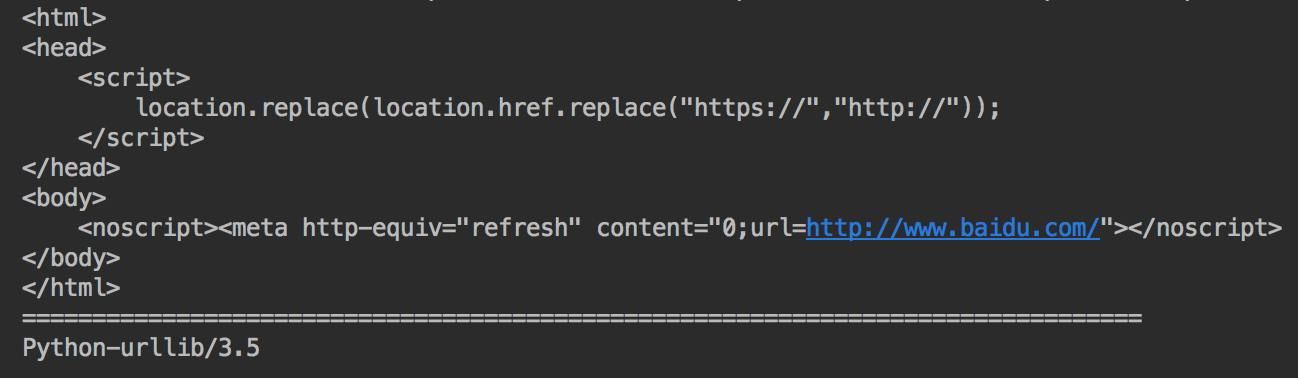




百度返回的内容和我们在浏览器上获得的完全不同。

* + 1. 案例 使用Request对象获取百度首页页面





换成使用Request对象也是相同结果。我们发现产生的Request对象的"User-agent"报头为"Python-urllib/3.5"。

* + 1. 设置headers中User-Agent的值来模仿浏览器的访问行为

宣称自己是什么类型的访问者，如某个版本的浏览器，等等。

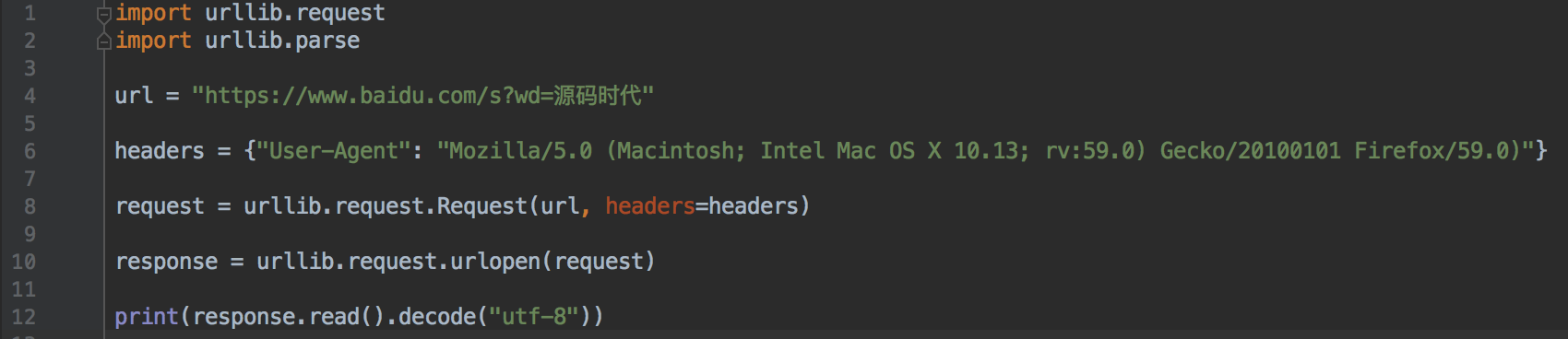


* 1. GET请求

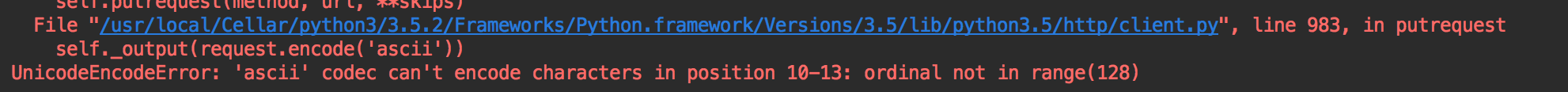
GET请求的所有参数都在URL地址中，通常用于我们向服务器获取数据。比如我们通过百度搜索“源码时代”，浏览器地址栏显示：<https://www.baidu.com/s?wd=源码时代>。

* + 1. 案例 尝试通过python代码获取“源码时代”百度搜索页面

如果我们试图在python程序中设定url为"https://www.baidu.com/s?wd=源码时代"，代码如下



则代码运行会出错，错误信息如下：

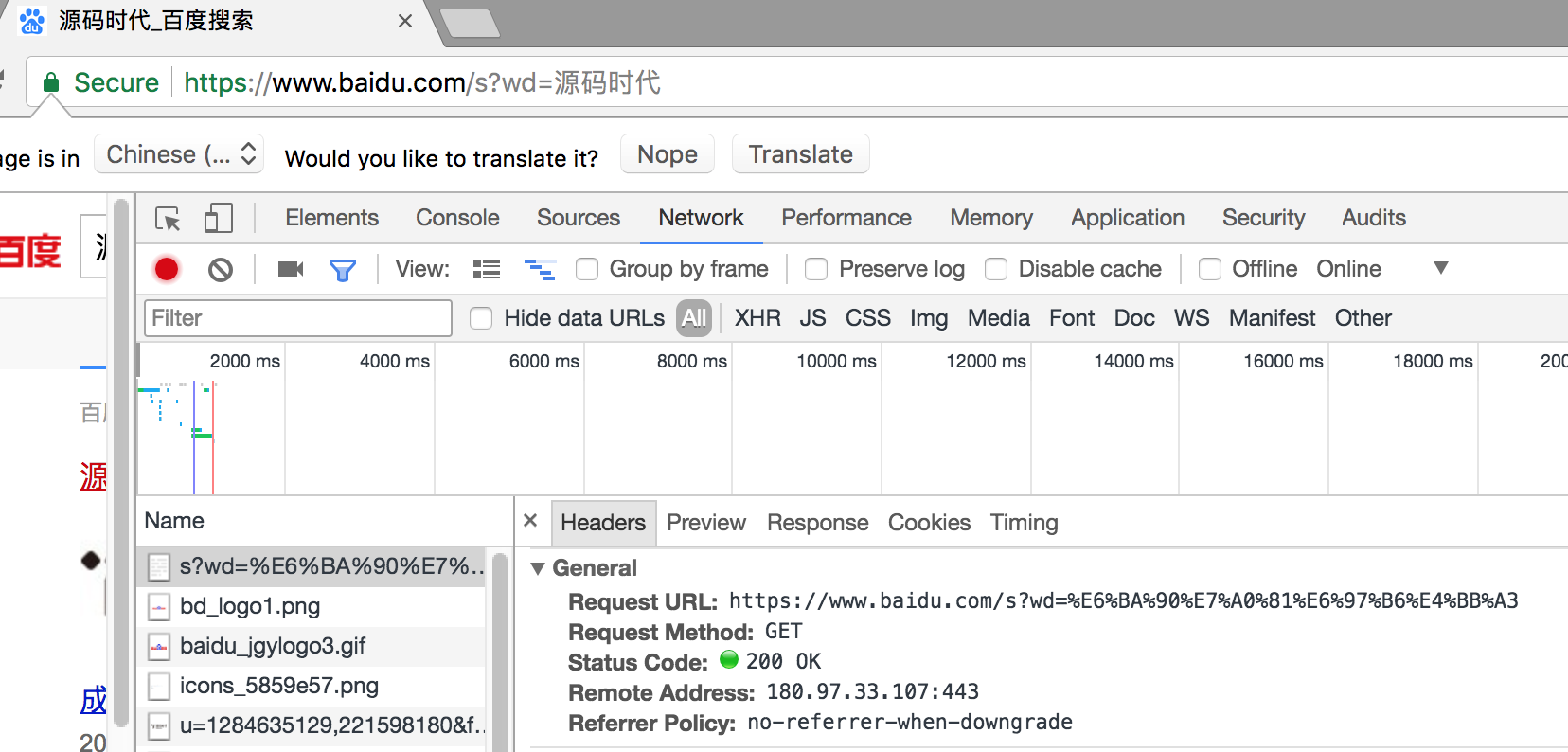


该错误信息指的是url字符串中的汉字不能编码为ASCII字符串，而HTTP/HTTPS协议只接受纯ASCII编码的URL地址。

因此我们只能通过某种方式，把汉字转换为HTTP/HTTPS协议接受的ASCII编码形式。

* + 1. 案例6 通过python代码获取“源码时代”百度搜索页面

事实上，我们在浏览器中做如下搜索时，我们复制浏览器地址栏将会得到"https://www.baidu.com/s?wd=%E6%BA%90%E7%A0%81%E6%97%B6%E4%BB%A3"。

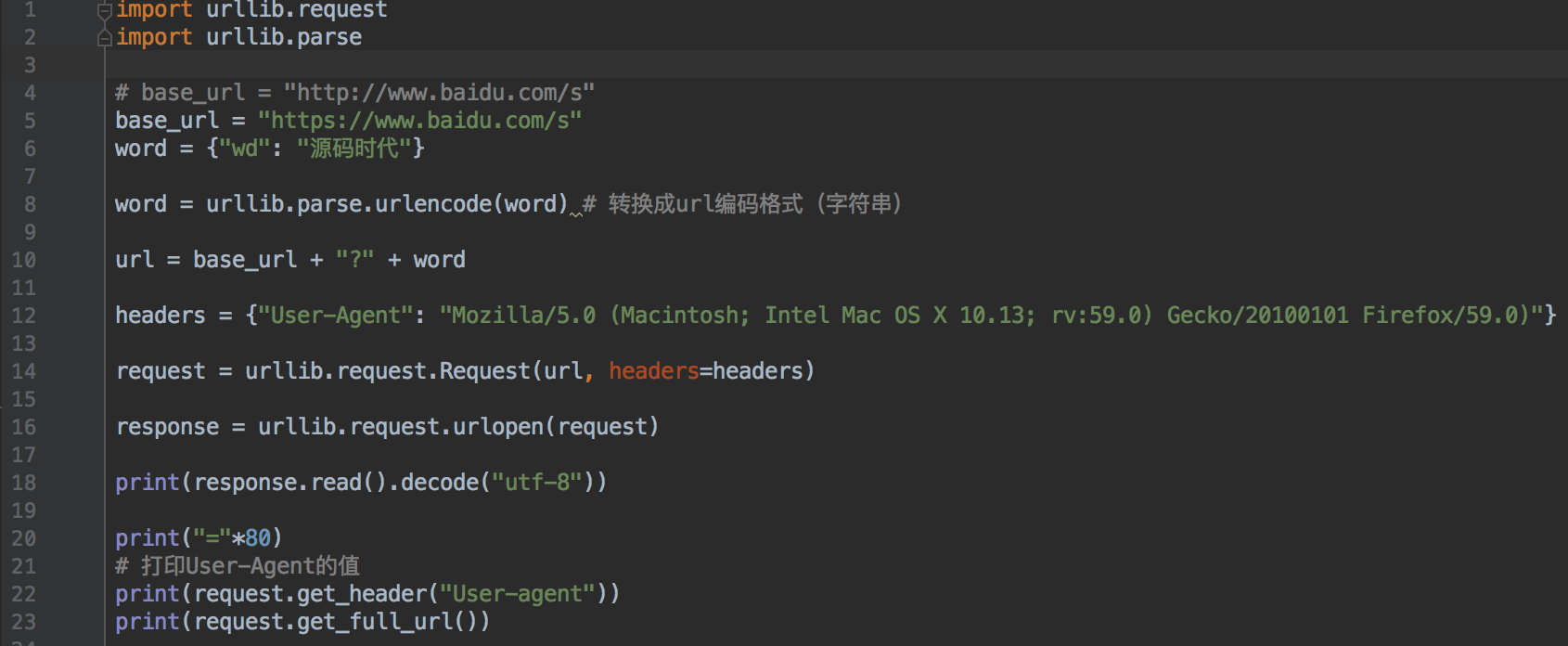


这是因为浏览器会自动将输入的url地址转换为"https://www.baidu.com/s?wd=%E6%BA%90%E7%A0%81%E6%97%B6%E4%BB%A3"。

在其中我们可以看到在请求部分里，http://www.baidu.com/s? 之后出现一个长长的字符串，其中就包含我们要查询的关键词源码时代，于是我们可以尝试用默认的Get方式来发送请求。

我们可以使用urllib.parse.urlencode(…)将<参数名:参数值>键值对的字典类型转换为编码后的参数串形式的ASCII字符串。

代码如下



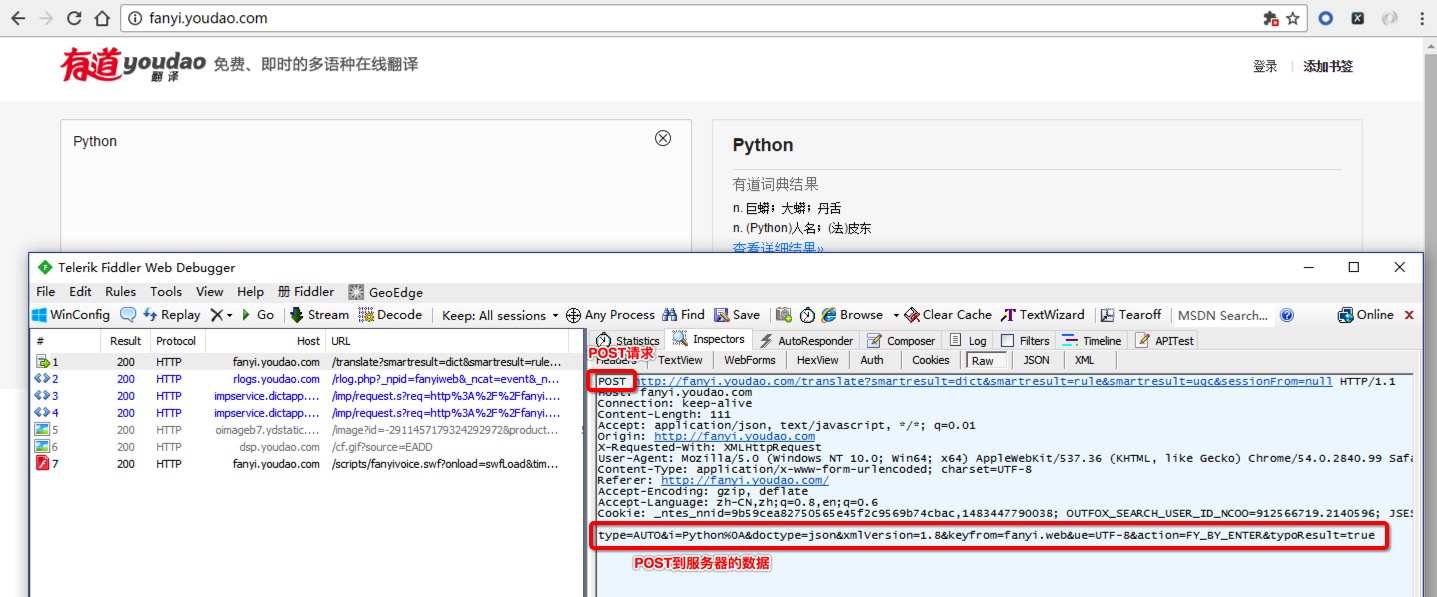
代码运行就会得到和浏览器一样的结果了。

* 1. POST请求

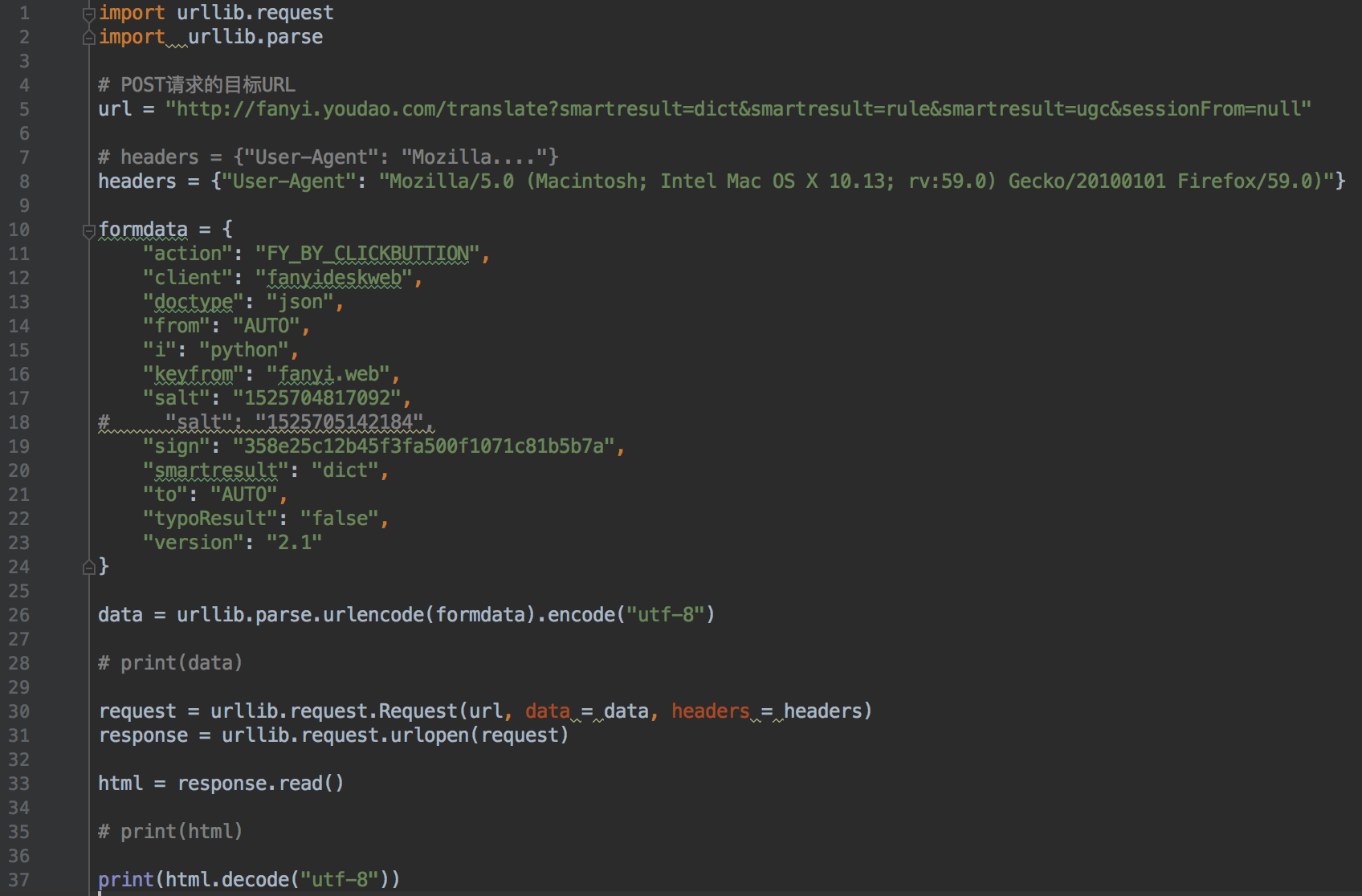
Request请求对象里面有data参数，它就是用在POST里的，我们要传送的数据就是这个参数data，这里的data是一个字典，里面要匹配键值对。

* + 1. 案例 网页版的有道词典翻译

输入测试数据，再通过使用Fiddler观察，其中有一条是POST请求，而向服务器发送的请求数据并不是在url里，我们可以试着模拟这个POST请求。



于是，我们可以尝试用POST方式发送请求。



* 1. Fiddler: HTTP代理工具
     1. 什么是Fiddler

Fiddler是一款强大Web调试工具，它能记录所有客户端和服务器的HTTP请求。Fiddler在服务器和浏览器之间当了个通信兵。

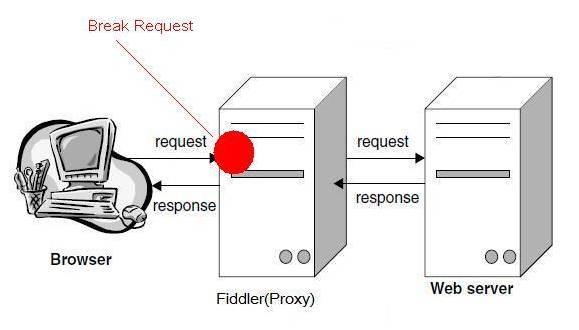
* + 1. 为什么使用Fiddler

Python程序中使用urllib进行的网络访问，我们想知道程序究竟发出什么请求，那么需要一种抓包工具，截取到程序发出的请求数据和得到的响应数据，进而分析程序实际行为，帮助我们进行开发和调试。

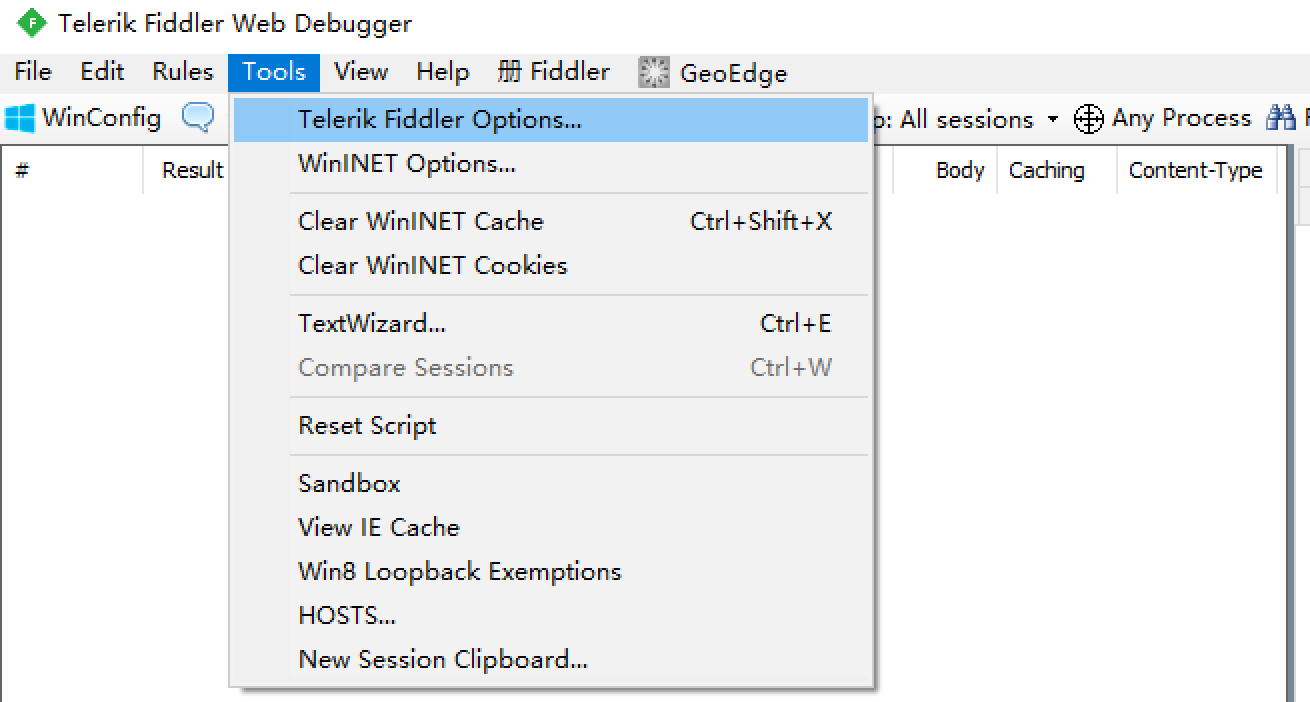
* + 1. 工作原理

Fiddler 是以代理web服务器的形式工作的，它被设置为监听127.0.0.1:8888

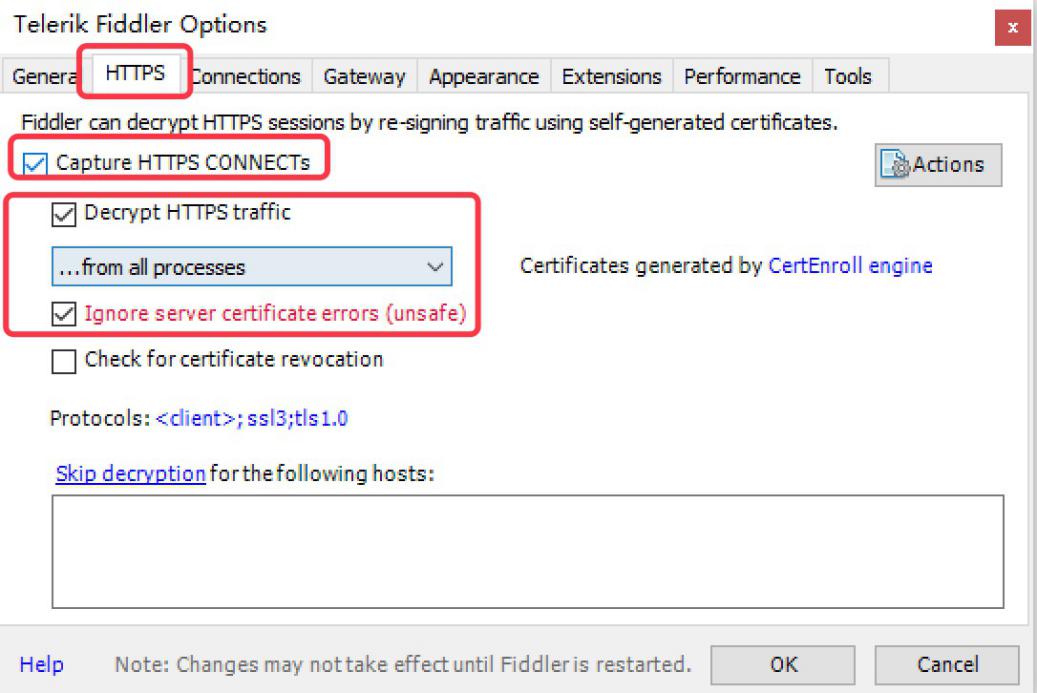
* + 1. Fiddler抓取HTTPS设置



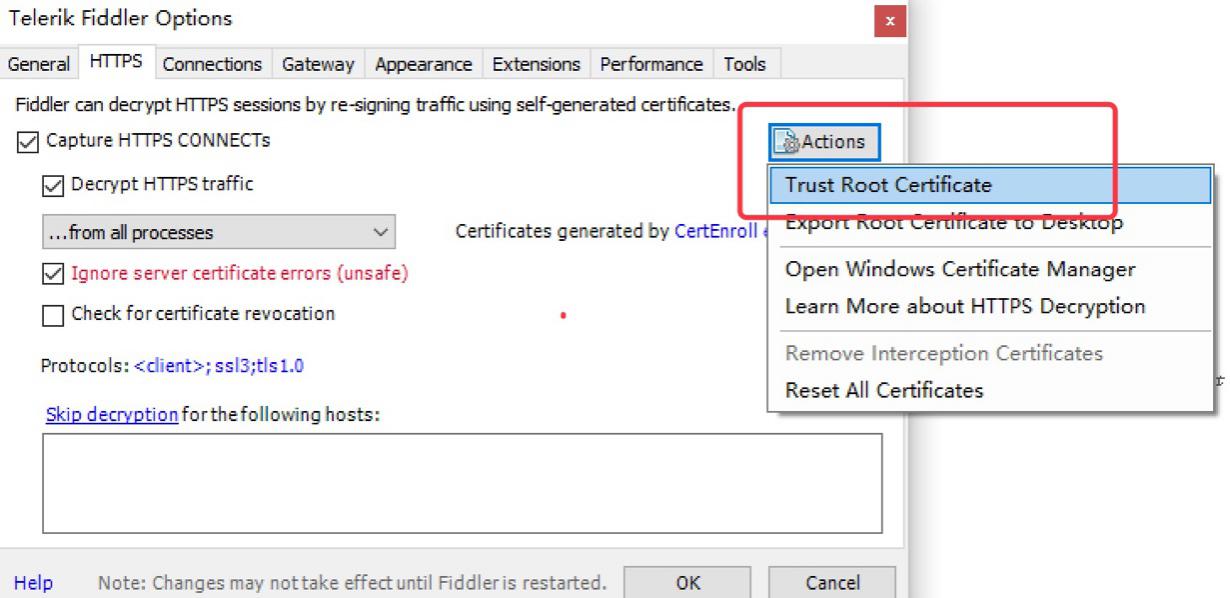
1. 启动Fiddler，打开菜单栏中的 Tools > Telerik Fiddler Options，打开“Fiddler Options”对话框。



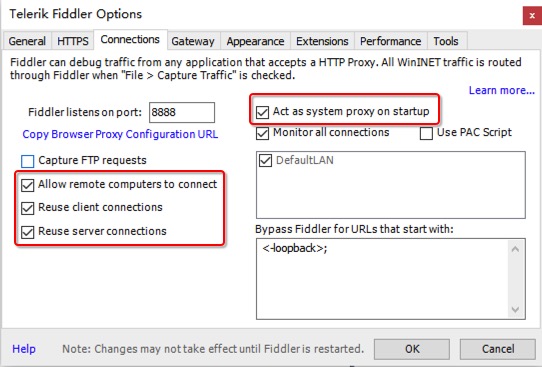
1. 对Fiddler进行设置：
   * 打开工具栏->Tools->Fiddler Options->HTTPS，
   * 选中Capture HTTPS CONNECTs (捕捉HTTPS连接)，
   * 选中Decrypt HTTPS traffic（解密HTTPS通信）
   * 另外我们要用Fiddler获取本机所有进程的HTTPS请求，所以中间的下拉菜单中选中...from all processes （从所有进程）
   * 选中下方Ignore server certificate errors（忽略服务器证书错误）



1. 为 Fiddler 配置Windows信任这个根证书解决安全警告：Trust Root Certificate（受信任的根证书）。



1. Fiddler 主菜单 Tools -> Fiddler Options…-> Connections
   * 选中Allow remote computers to connect（允许远程连接）
   * Act as system proxy on startup（作为系统启动代理）



1. 重启Fiddler，使配置生效（这一步很重要，必须做）。
   * 1. 对于部分Win 7系统可能会遇到的问题：

Fiddler 抓包https配置 提示creation of the root certificate was not successful 证书安装不成功，请在命令行执行如下命令：

* + - 1. 从命令行进入fiddler安装目录
      2. 执行如下命令

makecert.exe -r -ss my -n "CN=DO\_NOT\_TRUST\_FiddlerRoot, O=DO\_NOT\_TRUST, OU=Created by http://www.fiddler2.com" -sky signature -eku 1.3.6.1.5.5.7.3.1 -h 1 -cy authority -a sha1 -m 120 -b 11/11/2018

* + 1. Fiddler 如何捕获Chrome的会话

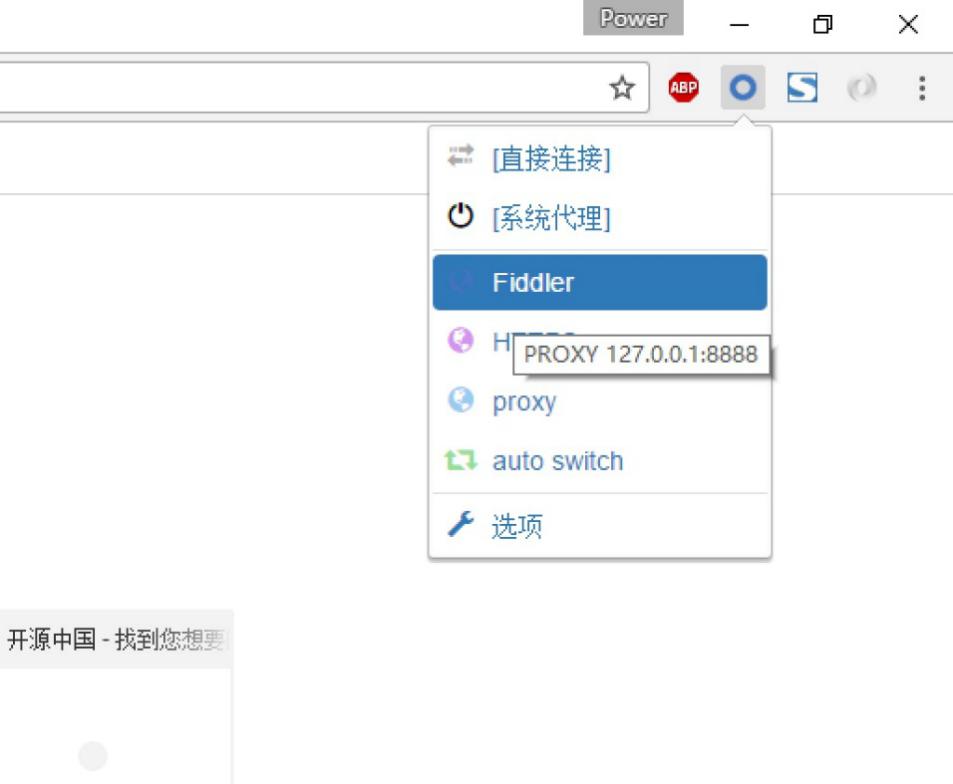
1. 安装SwitchyOmega 代理管理 Chrome 浏览器插件



1. 如图所示，设置代理服务器为127.0.0.1:8888

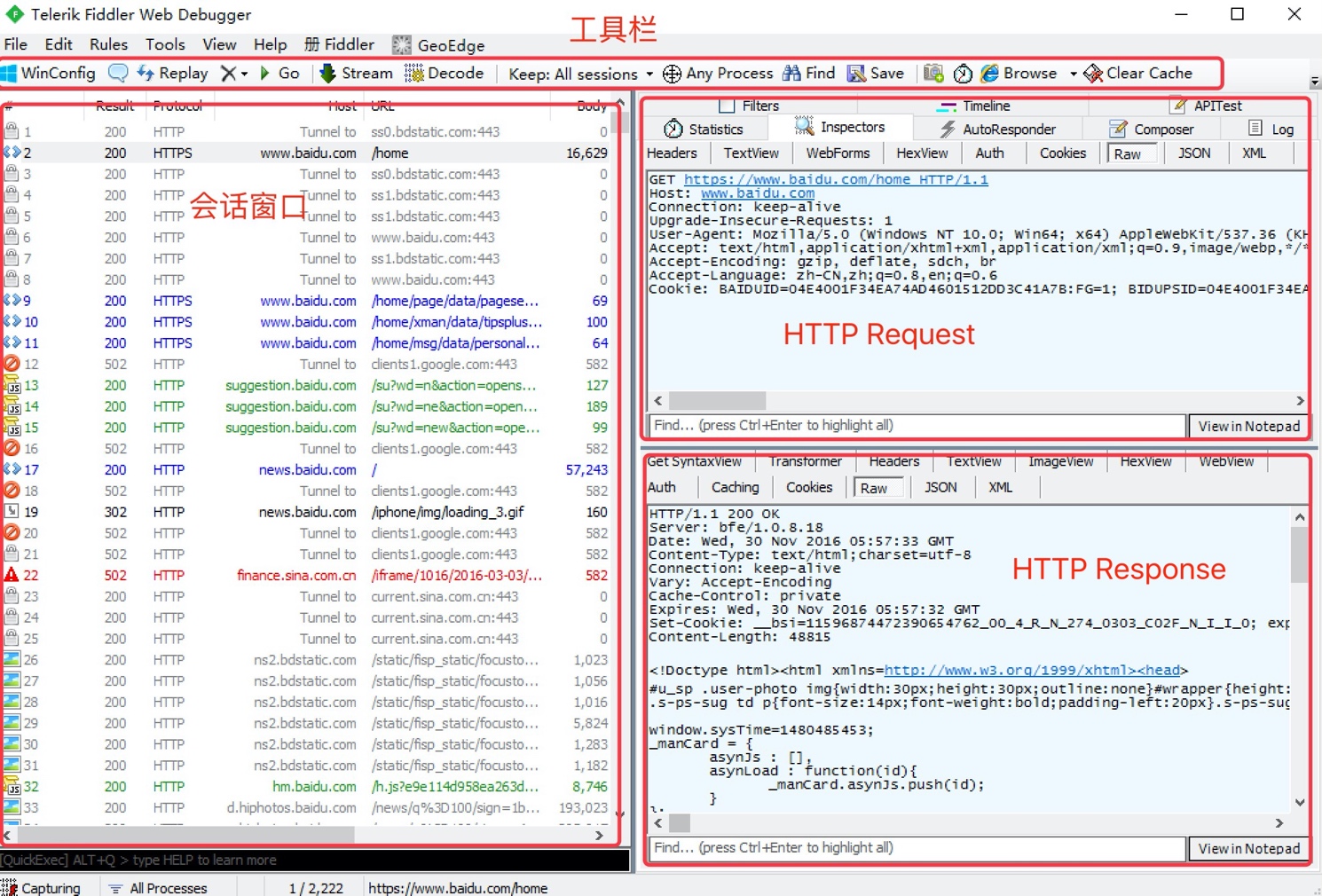


1. 通过浏览器插件切换为设置好的代理。



* + 1. Fiddler界面

设置好后，本机HTTP通信都会经过127.0.0.1:8888代理，也就会被Fiddler拦截到。



* + - 1. 请求 (Request) 部分详解

1. Headers —— 显示客户端发送到服务器的 HTTP 请求的 header，显示为一个分级视图，包含了 Web 客户端信息、Cookie、传输状态等。
2. Textview —— 显示 POST 请求的 body 部分为文本。
3. WebForms —— 显示请求的 GET 参数 和 POST body 内容。
4. HexView —— 用十六进制数据显示请求。
5. Auth —— 显示响应 header 中的 Proxy-Authorization(代理身份验证) 和 Authorization(授权) 信息.
6. Raw —— 将整个请求显示为纯文本。
7. JSON - 显示JSON格式文件。
8. XML —— 如果请求的 body 是 XML 格式，就是用分级的 XML 树来显示它。
   * + 1. 响应 (Response) 部分详解
9. Transformer —— 显示响应的编码信息。
10. Headers —— 用分级视图显示响应的 header。
11. TextView —— 使用文本显示相应的 body。
12. ImageVies —— 如果请求是图片资源，显示响应的图片。
13. HexView —— 用十六进制数据显示响应。
14. WebView —— 响应在 Web 浏览器中的预览效果。
15. Auth —— 显示响应 header 中的 Proxy-Authorization(代理身份验证) 和 Authorization(授权) 信息。
16. Caching —— 显示此请求的缓存信息。
17. Privacy —— 显示此请求的私密 (P3P) 信息。
18. Raw —— 将整个响应显示为纯文本。
19. JSON - 显示JSON格式文件。
20. XML —— 如果响应的 body 是 XML 格式，就是用分级的 XML 树来显示它 。

对于部分Win 7系统，[Fiddler 抓包https配置 提示creation of the root certificate was not successful 证书安装不成功](https://www.cnblogs.com/jayshsoft/p/7503171.html)，请在命令行执行如下命令：

* + - 1. 从命令行进入fiddler安装目录

<https://www.cnblogs.com/jayshsoft/p/7503171.html>

1. 课程总结
   1. 重点
      * 1. urllib标准库使用
        2. 请求报头中User-agent项的更改
        3. 中文网页的转码
   2. 难点
      * 1. 反爬虫机制的应对
        2. 签名，加盐等字段的生成
   3. 待抄写理论
      * 1. urllib.request.urlopen(…)函数的使用
        2. urllib.request.Request类的使用
        3. http.client.HTTPResponse类的使用
   4. 反爬虫应对机制阶段总结

目前阶段能用到的反爬虫机制：

* + - 1. 伪造headers，设置User-agent字段为常用浏览器的设置
      2. 降低访问频率，比如对于同一域名下不同页面，保证1秒以上的时间间隔
      3. 模拟普通用户的行为，比如对于同一域名下不同页面，每隔随机时间再爬取

后两条通常一同使用，比如对于同一域名下不同页面，每隔2~5秒的随机时间再爬取。

* 1. 排错技巧（避坑指南）

当爬虫类程序出现错误时：

* + - 1. 检查网络连接：包括网络物理连接状态和待爬取网页是否能在浏览器中显示
      2. 检查url地址是否正确,用浏览器打开
      3. 检查响应状态码，除了200以外的状态码，都查看其描述文档，分析问题可能原因
      4. 打印urlopen()函数返回的响应对象的响应体内容
      5. 检查headers的设置，特别是User-agent字段和Accept-Encoding字段，Accept-Encoding字段目前阶段一定要注释掉

1. 课后练习
   1. 必做题
      * 1. 实现上课代码，尽量领会后独立实现，包括自己组织语言做简要的注释
        2. 通过python程序爬取豆瓣网站Top250电影页面，将每页结果分别保存在文件中

豆瓣Top250电影页面第一页的URL是<https://movie.douban.com/top250?start=0&filter>=，第二页URL是<https://movie.douban.com/top250?start=25&filter>=，以此类推。

* 1. 选做题
     + 1. 从键盘输入出发地和目的地，爬取12306网站上当天的余票信息，将每条车次信息组织成字典，然后保存至文件中。(获取城市名对应的城市代码是一个难点；而应对反爬虫机制是另一个难点)

1. 面试题
   1. 常见的反爬虫应对机制有哪些？
   2. http、https协议有什么区别？
   3. robots协议是什么？
   4. 域名和IP之间有什么关系?
   5. POST、GET有什么区别？
2. 每日一练

复习网络的相关内容，重点是HTTP/HTTPS协议。