本文介绍使用Python的urllib2和cookielib，登录哈工大统一身份认证系统。

2017-3-16

爬网站的时候，很多都需要先登录。登录分为带验证码和不带验证码，不带验证码很简单，直接Post表单即可，带验证码需要获取验证码图片，使用某种手段解析出来，再带验证码提交。下面以登陆哈工大统一身份认证系统为例，解释如何完成一次带验证码的自动登录。对登录其他网站也有借鉴意义。

1. 分析网站信息：

【1】哈工大采用统一身份认证，即不论登录任何系统，都会跳转到统一身份认证页面进行认证，成功之后再跳转进对应页面。

研究生院转统一身份认证：

<http://ids.hit.edu.cn/authserver/login?service=http%3A%2F%2F219.217.227.152%3A8080%2Fhitgsmis%2FindexActionCAS.do>

教务处转统一身份认证：

<http://ids.hit.edu.cn/authserver/login?service=http%3A%2F%2Fjwts.hit.edu.cn%2FloginCAS>

乐学网转统一身份认证：

<https://ids.hit.edu.cn/authserver/login?service=https%3A%2F%2Fcms.hit.edu.cn%2Flogin%2Findex.php%3FauthCAS%3DCAS>

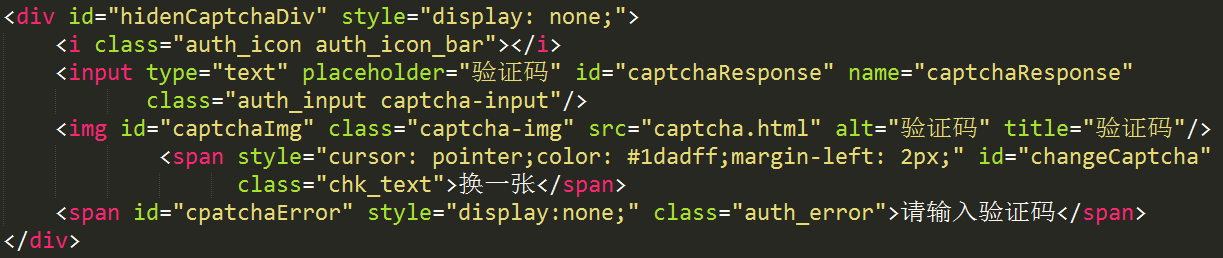
可以看到，它是用service值来标记从哪来，认证成功之后要到哪去。

【2】网页完整信息见：哈工大统一身份认证.html

【3】使用审查元素，可以找到登录表单，发现它长这样：



还可以找到验证码输入框，发现它长这样：

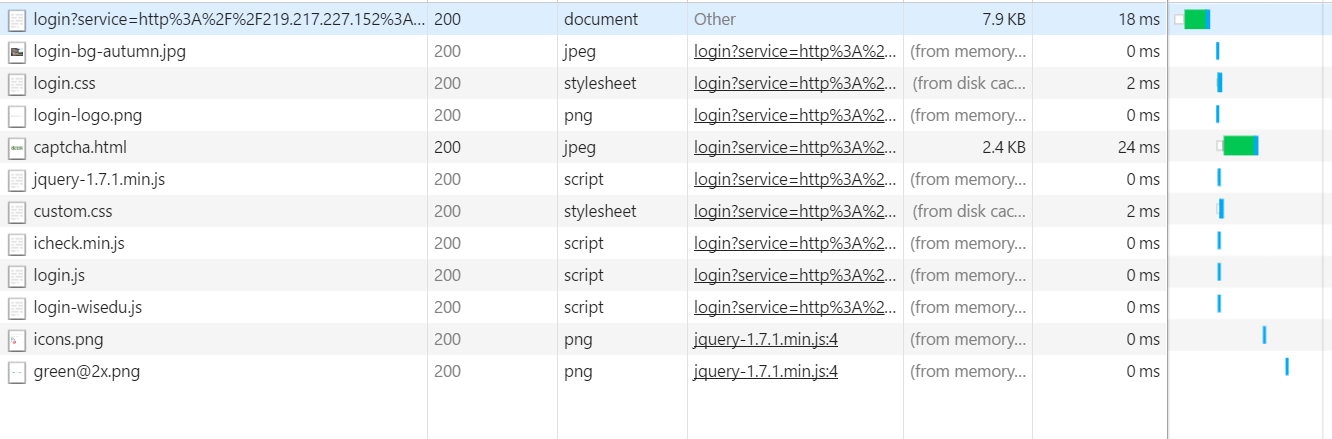


这里面有三个特别重要的信息，分别是：form的action；type为hidden的神秘input；验证码的src。其内容稍后分解。

【4】用python模拟登录，最好先看看浏览器是怎么登陆的。Ctrl+Shift+I，呼出Chrome的强大控制台，点击Network选项卡，刷新页面，观察浏览器工作过程。

下面是统一身份认证界面的加载流程，第一项Login是我要open的页面，因为登录只需要提交表单，其他的js，css可以忽略了。

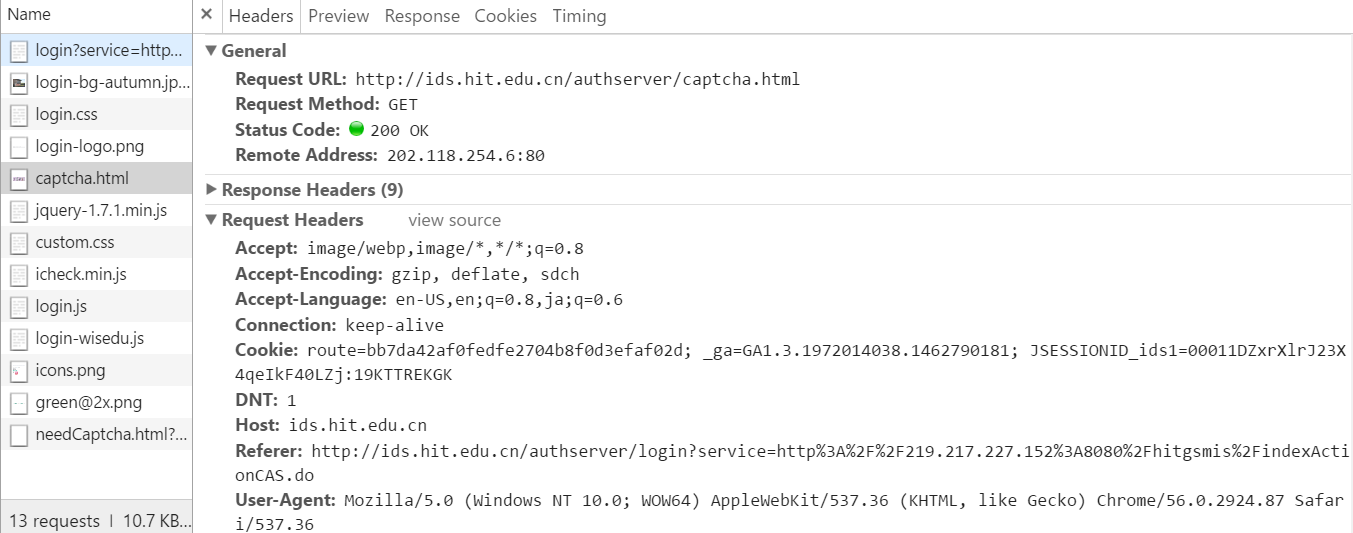
浏览器做了这么几件事：1.获取login页面(就是url指向的页面)；2.获取login引用的各种资源：就是下面那一堆。



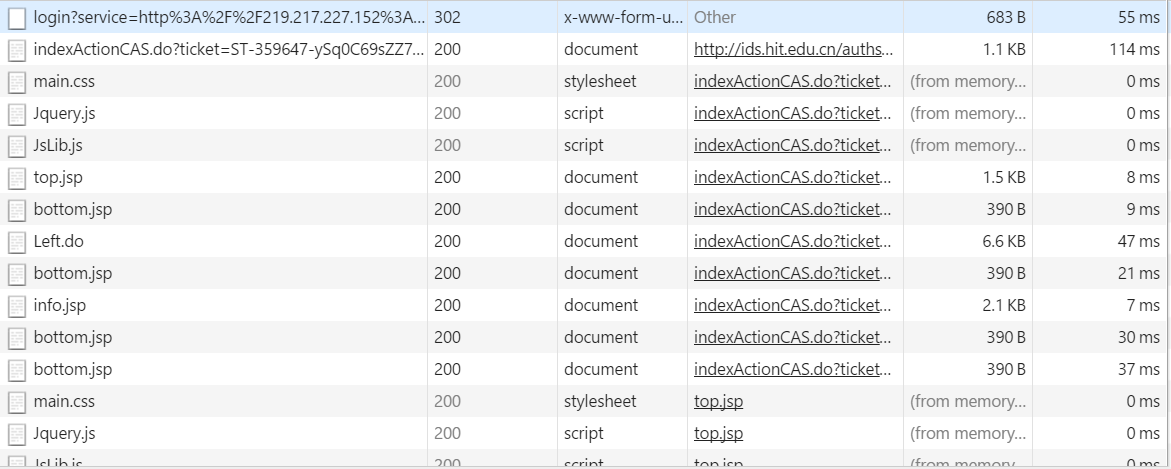
我们的第一步就是获取login页面，下面点开左侧打开详细信息，可以看到login的RequestHeaders。所以我去打开统一认证页面的时候，就使用一样的头（貌似只要有User-Agent那项就行，保险我还是都加上了）。



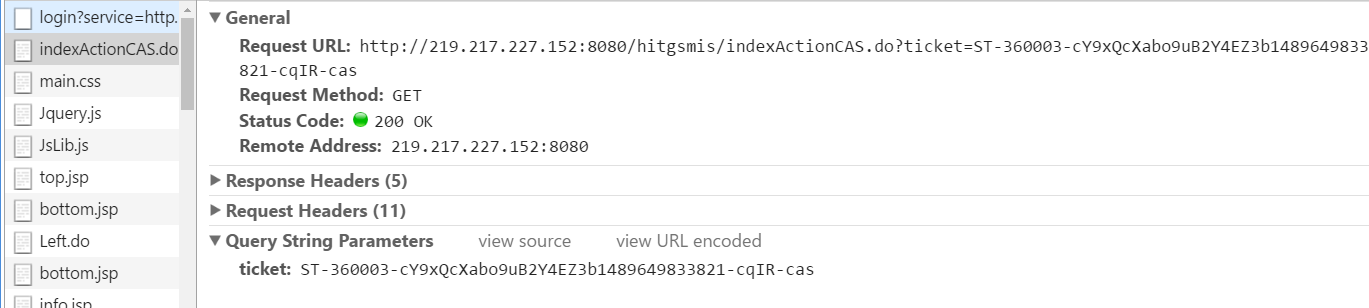
captcha.html是验证码页面，request的时候头与login不同，但使用login的头也没关系。captcha.html非常简单，没有引用任何其他资源，就是简单Get，返回一张验证码图片。从右侧栏中可以看到，只由General，Response Headers和Request Headers三项。



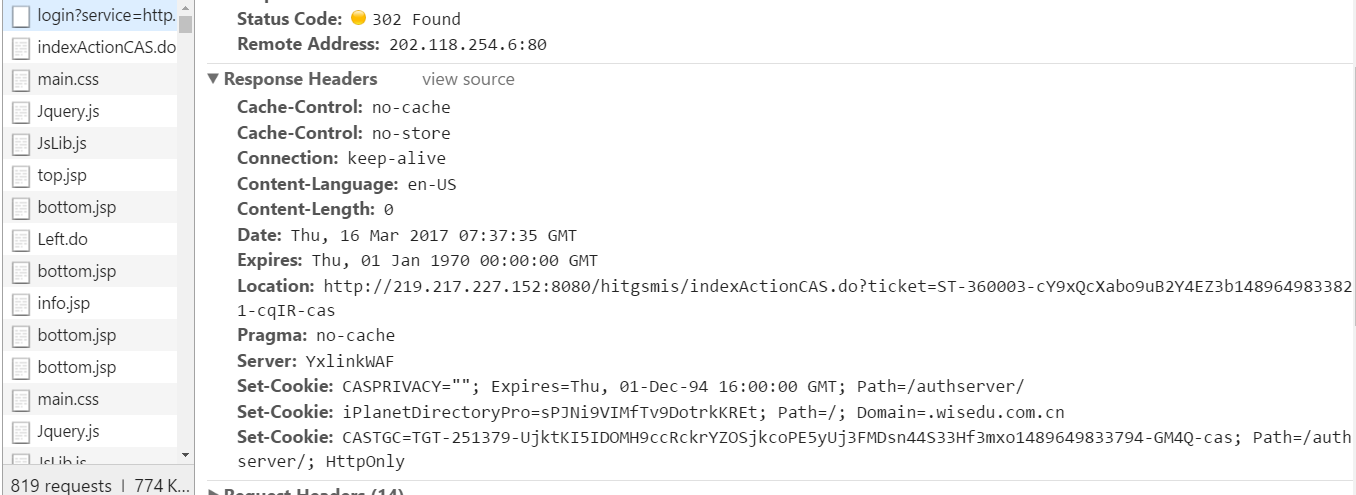
下面是点击登录后，浏览器的执行过程。过程很麻烦，第一个是form表单提交，302表示被重定向到了第二个url。第二项是Get了具体的页面，如下图，因此可以推测，在第一个页面完成身份验证后，会根据【1】中的service不同重定向到对应页面。



浏览器完整过程



第二项的General信息：第一项被重定向到对应的页面（第二项），第二项Get了具体页面

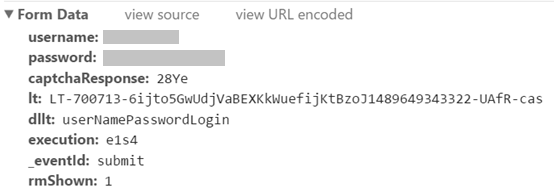


第一项的response headers：这里包含了重定向信息？

可见身份验证工作由第一步完成，与后面无关。打开第一步详细，可见Post方法提交表单，request url是表单里的action。滚轮向下看一下FormData部分，可以看到表单提交的信息。



提交Form的General信息



Form的Data信息

【5】流程追踪到此结束，总结一下就是：

1. 向login界面发送头为XXX的Get请求，获取页面。
2. 获取captcha.html，解析验证码。
3. 输入用户名，密码，验证码，Post给login表单中的action。

看起来很简单吧~

二.实际操作

【1】关于验证码的小秘密

captcha.html不限制访问，且只返回验证码，每次Get内容都不相同。说明这是专门用于申请验证码的页面，但是它是怎么把验证码和登录人的输入关联的？还记得之前奇怪的type=hidden的input标签吗？仔细看Form Data信息图中，它们是不是都在那里。它们是： 'lt','dllt','execution','\_eventId','rmShown'。

我做实验后发现很可能是这样：

1.浏览器Get login，服务端返回一个带这些标记字段的页面，标记写在type=hidden的input标签中。

2.服务端下一个被请求的验证码，会和最近一次返回的login页面上的标记字段对应。

3.页面Post表单，加入了label信息。服务端根据表单中的标记去找对应验证码，再看匹配不匹配。(很可能是lt标记，因为cas表示身份认证)

缺点是坏人一直刷新验证码，会导致映射错误；网络过慢会导致映射错误。

1. 解决方案设计
2. 向login界面发送头为XXX的Get请求，获取页面内容content。
3. 从中解析出'lt','dllt','execution','\_eventId','rmShown'它们对应的值，叫做label\_values。
4. 向captcha.html发送Get，获取页面内容，存入本地，并打开。
5. 等待用户输入username，password，captcha。
6. 根据用户输入和2中的label\_values构造表单，提交给Form中action指定的url。
7. 登录成功，查看cookie，发现多了获取授权的cookie了。将登陆成功的openner和cj返回给调用者，由调用者使用，可以畅通无阻的访问网站了。

【3】代码实现见LoginHIT.py。