综合训练–抓取91看剧的视频

首先, 我们先从一个简单的案例入手.

我们想要抓取网上的视频资源就必须要了解我们的视频网站是如何工作的. 这里我用91看剧来做举例. 其他网站的原理是一样的.

1.视频网站是如何工作的

假设, 你现在想要做一个视频网站. 也有很多的UP猪帮你上传视频. OK, 作为服务器作者的你. 只需要把视频保存起来. 然后给出一个视频的链接即可. 然后在你的HTML代码中通过 `video` 标签引入即可.

4_2_多线程.mp4	2021/3/3 下午4:34	704.7 MB	Monosnap vide
4_1_第四章概述.mp4	2021/3/2 下午7:35	192.3 MB	Monosnap vide
1_7_补充_关闭resp.mp4	2021/3/2 下午5:52	169.4 MB	Monosnap vide
3_5_综合训练_抓取网易云音乐评论信息.mp4	2021/3/2 下午5:43	4.82 GB	Monosnap vide
3_3_防盗链_抓取梨视频.mp4	2021/3/1 下午9:02	1.89 GB	Monosnap vide
3_4_代理.mp4	2021/3/1 下午5:56	544.6 MB	Monosnap vide
3_2_处理cookie_登录小说网.mp4	2021/2/26 下午6:51	1.23 GB	Monosnap vide
3_1_requests进阶概述.mp4	2021/2/26 下午6:20	116.7 MB	Monosnap vide
2_9_xpath实战_抓取猪八戒网信息.mp4	2021/2/26 下午4:19	2.57 GB	Monosnap vide
2_8_xpath入门_02.mp4	2021/2/25 下午7:23	959.7 MB	Monosnap vide
2_8_xpath入门_01.mp4	2021/2/25 下午6:07	876.4 MB	Monosnap vide
2_7_bs4解析案例-抓取优美图库图片.mp4	2021/2/25 下午4:59	2.03 GB	Monosnap vide
2_6_bs4解析入门-搞搞菜价.mp4	2021/2/24 下午7:24	1.42 GB	Monosnap vide
2_5_bs4解析前戏-html语法规则.mp4	2021/2/24 下午6:48	549.9 MB	Monosnap vide
2_4_屠戮盗版天堂电影信息.mp4	2021/2/24 下午5:16	2.94 GB	Monosnap vide
2_4_手刃豆瓣TOP250电影排行.mp4	2021/2/24 下午4:26	1.16 GB	Monosnap vide
2_3_python的re模块使用.mp4	2021/2/23 下午8:09	1.58 GB	Monosnap vide
2_2_re解析_正则表达式_02.mp4	2021/2/23 下午6:16	763 MB	Monosnap vide
2_2_re解析_正则表达式_01.mp4	2021/2/23 下午5:45	967.9 MB	Monosnap vide
2_1_数据解析概述.mp4	2021/2/23 下午5:19	171.2 MB	Monosnap vide
1_6_3_requests入门.mp4	2021/2/23 下午4:51	1.1 GB	Monosnap vide
1_6_2_requests入门.mp4	2021/2/23 下午4:29	373.7 MB	Monosnap vide
1_6_1_requests入门.mp4	2021/2/23 下午4:12	1.18 GB	Monosnap vide
1_5_http协议.mp4	2021/2/22 下午4:45	753.6 MB	Monosnap vide
1_4_web请求过程剖析.mp4	2021/2/20 下午6:45	1.22 GB	Monosnap vide
1_3_手刃一个小爬虫.mp4	2021/2/20 下午4:03	872.2 MB	Monosnap vide
1_2_本课程使用的软件.mp4	2021/2/19 下午4:39	610.3 MB	Monosnap vide
1_1_爬虫概述.mp4	2021/2/19 下午3:36	492.6 MB	Monosnap vide

```
1 <video src="1_1_爬虫概述.mp4"></video>
```

就可以了. 但是, 如果你这么做. 你的用户和老板一定会把你骂的狗血临头. 为什么呢?

假设你的视频是10个G的高清无码大资源. 那么此时, 你的用户和你老板将面临如下困境

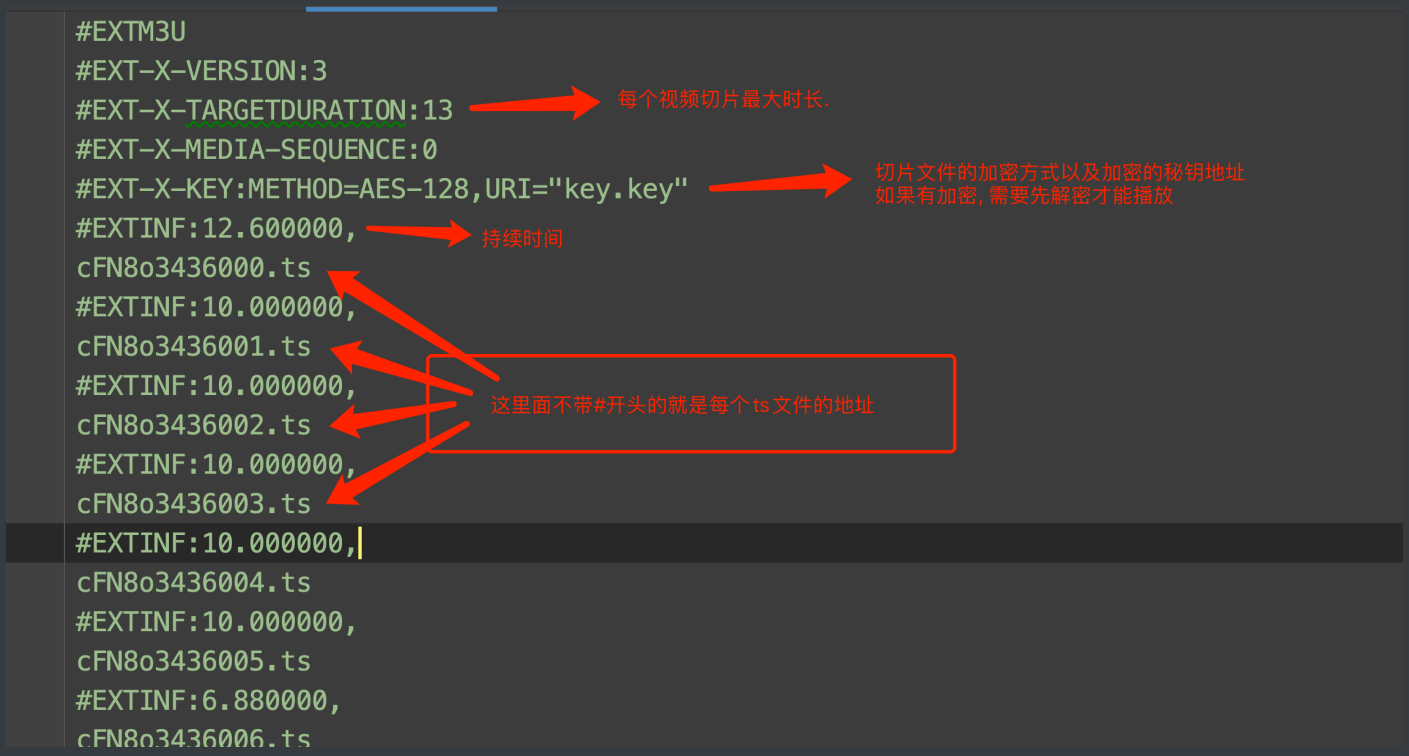
1. 用户: 这个视频怎么加载的这么慢. 点击快进也快进不了. 太慢了. 塔喵的烦死了.
2. 老板: 怎么这个月的流量费又这么高啊. 要死的拉好不~

为什么会这样? 聪明的我告诉你答案. 你的视频那么大. 每次用户打开的时候. 可能只是差了最后几分钟没看呢. 那此时它必须把整个视频都传输完毕. 才能看到他想看的那里. 等待时间肯定超长的好不. 而每次都要把10G的文件进行网络传输. 流量费~你懂的. 三大运营商最喜欢的就是你这种朴实无华的送钱行为.

OK~ 不扯了. 但凡有点儿经验的程序员肯定会想办法把用户上传好的视频进行转码(不同清晰度)做切片(ts)处理. 这样既方便用户进行大跨度的调整进度条(最小延迟). 也能为公司节省大量的流量费.

既然要把视频切成非常多个小碎片. 那就需要有个文件来记录这些小碎片的路径. 该文件一般为M3U文件. M3U文件中的内容经过UTF-8的编码后, 就是M3U8文件. 今天, 我们看到的各大视频网站平台使用的几乎都是M3U8文件.

如何解读M3U8文件.



基本知道这些就够了.

2. 先来个简单的试试

目标: 哲仁皇后. 走起

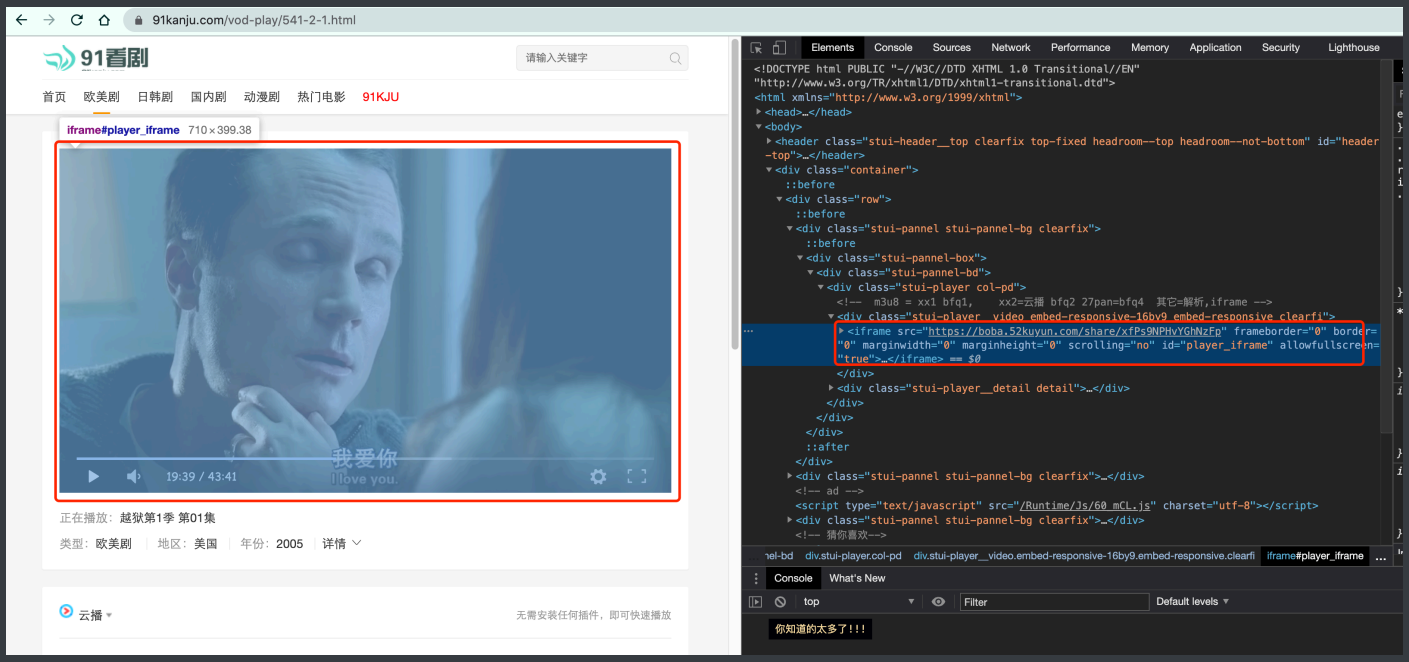
```
import requests
import re

headers = {
    "user-agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.192 Safari/537.36"
}

url = "https://www.91kanju.com/vod-play/54812-1-1.html"
resp = requests.get(url, headers=headers)

obj = re.compile(r"url: '(?P<url>.*?)',", re.S)
result = obj.search(resp.text)
resp.close()

url = result.group('url')
```

```
resp2 = requests.get(url, headers=headers)
f = open("哲仁王后.m3u8", mode="wb")
f.write(resp2.content)
resp2.close()
f.close()

# 读取m3u8视频文件
with open("哲仁皇后.m3u8", mode="r", encoding="utf-8") as f:
    count = 1
    for line in f:
        line = line.strip()
        if line.startswith("#"):
            continue

        resp3 = requests.get(line, headers=headers)
        with open(f"video2/{count}.ts", mode="wb") as ff:
            ff.write(resp3.content)
        resp3.close()
        count += 1
        print("完事儿1个")
        if count >= 10:  # 测试. 别整太过分
            break

```

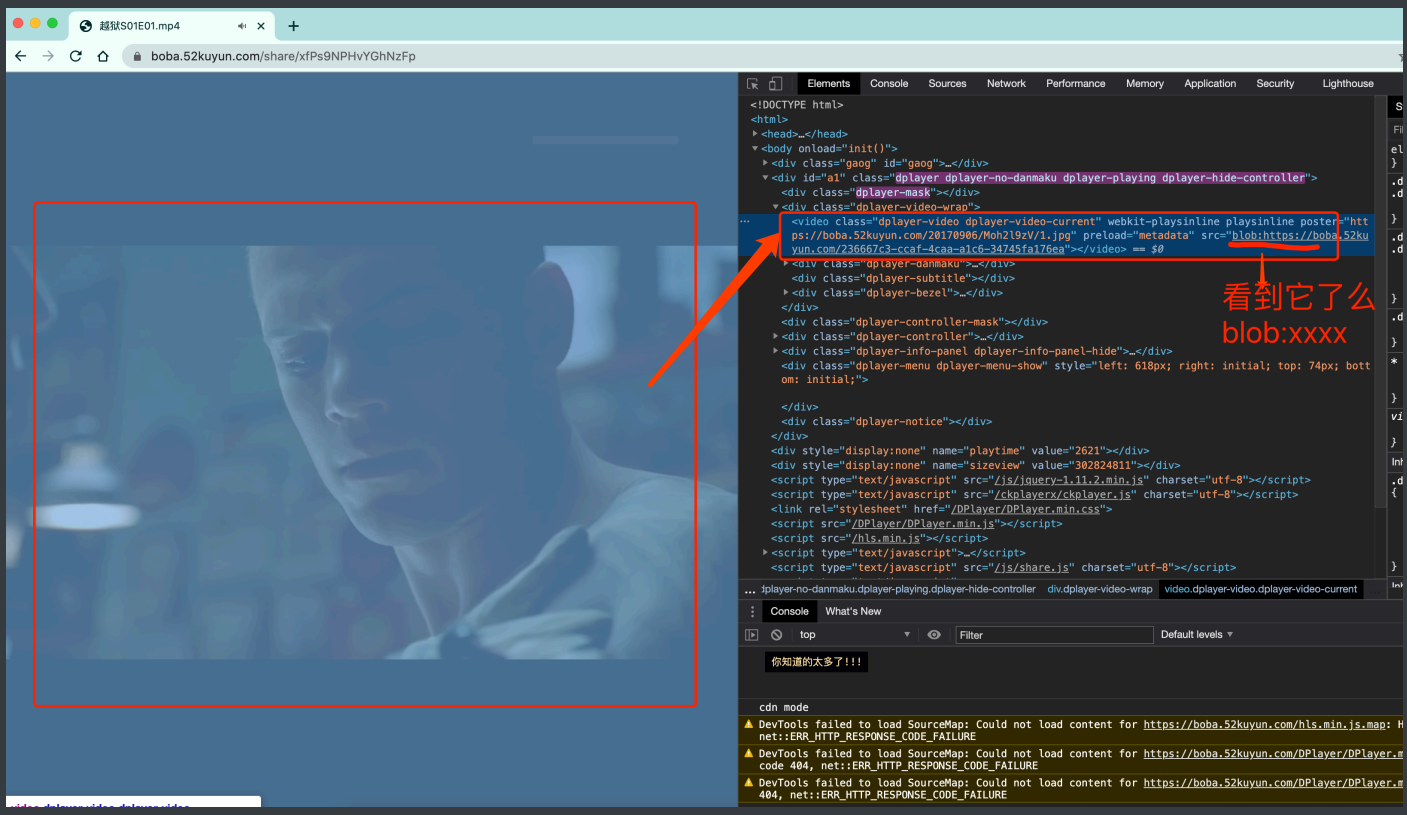
```
# 最后，需要用quick time把视频拼接起来.
# 看到了么. 上面那个居然没有用协程，也没有多线程. 为什么呢? 因为太简单了. 我们来个复杂的.
```

3. 真正的挑战. 抓取<越狱>视频第一集

首先, 盯紧我们的目标, 你会发现, 我们想要的视频被放在了一个iframe里面



我们打开这个iframe的链接, 发现这个链接里嵌套了一个播放器.



而这个播放器中的视频地址是一个blob:xxxx, 这是个什么鬼. blob:https并不是一种协议， 而是html5中blob对象在赋给video标签后生成的一串标记. 说简单点儿. 就是防止爬虫能直接拿到视频下载地址, 中间倒了一手而已.

此路不通, 我们再去观察一下这个页面的页面源代码.

```
            </div>
        </div>

        <div id="a1"></div>

        <div style="display:none" name="playtime" value="2621"></div>
        <div style="display:none" name="sizeview" value="302824811"></div>
        <script type="text/javascript" src="/js/jquery-1.11.2.min.js" charset="utf-8"></script>
        <script type="text/javascript" src="/ckplayerx/ckplayer.js" charset="utf-8"></script>
                <link rel="stylesheet" href="/DPlayer/DPlayer.min.css">
        <script src="/DPlayer/DPlayer.min.js"></script>
                <script src="/hls.min.js"></script>
                        <script type="text/javascript">
            var video_player= 'dplayer'
            var tracker_url = ''
            var signaler_url = ''
            var hosts = '';
            var redirecturl = "http://vip.okzybo.com";
            var videoid = "xfPs9NPHvYGhNzFp";

            var id = 'xfPs9NPHvYGhNzFp'
            var     l = ''
            var     r= ''
            var     t= '15'
            var     d= ''
            var     u= ''

            var main = "/20170906/Moh2l9zV/index.m3u8?sign=548ae366a075f0f9e7c76af215aa18e1";
            var xml = "/20170906/Moh2l9zV/index.xml?sign=548ae366a075f0f9e7c76af215aa18e1";
            var pic = "/20170906/Moh2l9zV/1.jpg";
            var thumbnails = "/20170906/Moh2l9zV/thumbnails.jpg";
        </script>
        <script type="text/javascript" src="/js/share.js" charset="utf-8"></script>
        <script type="text/javascript">

        </script>

    <!--/广告添加区域-->
</body>

</html>
```

抓取这个视频的整体思路:

1 拿到主页的页面源代码. 获取到iframe的链接
2 拿到iframe链接的页面源代码. 获取到m3u8链接
3 下载m3u8文件(2层),稍显繁琐.
4 对文件进行检索, 拿到小视频的链接,进行异步下载
5 解密合并小的切片文件还原为大的MP4文件进行播放

并且, 我们在浏览器的network中也能非常清晰的看到整个浏览器的请求过程, 几乎和我们刚才总结的思路是一致的 接下来, 上代码吧. 我依然会分步骤的去描述如何抓取到最后的视频文件

```python
import requests
from bs4 import BeautifulSoup
import re
```

```python
import asyncio
import aiohttp
import aiofiles
from Crypto.Cipher import AES
import os

def get_iframe_url(url):
    resp = requests.get(url)
    main_page = BeautifulSoup(resp.text, "html.parser")
    iframe = main_page.find("iframe")
    resp.close()
    return iframe.get("src")


def get_first_m3u8_url(iframe_url):
    iframe_resp = requests.get(iframe_url)
    obj = re.compile(r'var main = "(?P<first_m3u8>.*?)"', re.S)
    result = obj.search(iframe_resp.text)
    iframe_resp.close()
    return result.group('first_m3u8').strip()


def download_m3u8_file(url, name):
    resp = requests.get(url)
```

```
    resp.encoding='utf-8'
    with open(name, mode="w", encoding="utf-8") as f:
        f.write(resp.text)
    resp.close()


async def download_ts(session, url, name):
    async with session.get(url) as resp:
        async with aiofiles.open(f"video3/{name}", mode="wb") as f:
            await f.write(await resp.content.read())
    print(f"{name}~OK!")


async def aio_download(up_url):
    tasks = []
    async with aiofiles.open("第二层_tmp_m3u8.txt", mode="r", encoding="utf-8") as f:
        async with aiohttp.ClientSession() as session:
            async for line in f:
                if line.startswith("#"):
                    continue
```

```
48                  line = line.strip()
49                  ts_url = up_url + "/" + line
50                  name = line
51                  # 将异步任务添加到列表
52
    tasks.append(download_ts(session, ts_url,
    name))
53              # 开炮!!
54              await asyncio.wait(tasks)
55
56
57  def get_key(url):
58      resp = requests.get(url)
59      resp.encoding='utf-8'
60      key = resp.text
61      resp.close()
62      return key
63
64
65  async def dec_ts(name, key):
66      aes = AES.new(key=key,
    IV=b"0000000000000000", mode=AES.MODE_CBC)
67      async with aiofiles.open("video3" + "/" +
    name, mode="rb") as f1,\
68              aiofiles.open("video4" + "/" +
    "tmp_" + name, mode="wb") as f2:
```

```
69
70          bs = await f1.read()
71          await f2.write(aes.decrypt(bs))
72      print(f"{name}~解密OK~")
73
74
75  async def aio_dec(key):
76      tasks = []
77      async with aiofiles.open("第二层
    _tmp_m3u8.txt", mode="r", encoding="utf-8") as
    f:
78          async for line in f:
79              if line.startswith("#"):
80                  continue
81              line = line.strip()
82              tasks.append(dec_ts(line, key))
83          # 异步解密
84          await asyncio.wait(tasks)
85
86
87  def merge_ts():
88      s = []
89      with open("第二层_tmp_m3u8.txt") as f:
90          for line in f:
91              if line.startswith("#"):
92                  continue
```

```
            line = line.strip()
            s.append("./video4/tmp_"+line)
    names = " ".join(s)
    os.system(f"cat  {names} > movie.mp4")

# 主程序
def main(url):
    iframe_url = get_iframe_url(url)
    # 1. iframe的域名
    iframe_domain = iframe_url.split("/share/")[0]
    # 2. 拿到m3u8地址(顶级)
    first_m3u8_url = get_first_m3u8_url(iframe_url)
    # 3.1 下载first_m3u8
    first_m3u8_url = iframe_domain + first_m3u8_url
    download_m3u8_file(first_m3u8_url, "顶层_tmp_m3u8.txt")
    # 3.2 拿到m3u8的上层url, 目的是拼接第二层m3u8的url以及后面的key.key等等
    first_m3u8_url_up = first_m3u8_url.rsplit("/", 1)[0]
    # 3.3 下载第二层m3u8文件
    with open("顶层_tmp_m3u8.txt", mode="r", encoding='utf-8') as f:
```

```
112         for line in f:
113             if line.startswith("#"):
114                 continue
115             else:
116                 second_m3u8_url =
    first_m3u8_url_up + "/" + line.strip()
117
118     download_m3u8_file(second_m3u8_url, "第二层
    _tmp_m3u8.txt")
119     # 4. 读取第二层m3u8文件. 开始下载ts
120     second_m3u8_url_up =
    second_m3u8_url.rsplit("/", 1)[0]
121     # 异步下载
122
     asyncio.run(aio_download(second_m3u8_url_up))
     # 测试的时候慎重啊
123     # 5. 解密与合并
124     # 5.1. 拿到key
125     key = get_key(second_m3u8_url_up +
    "/key.key")  # 这里就不去m3u8里找了.偷个懒
126     # print(key)
127     # key = "c5878c26baaaac8c"
128     # 5.2. 解密
129     asyncio.run(aio_dec(key))
130     # 5.3. 合并
131     merge_ts()
```

```
132
133
134 if __name__ == '__main__':
135     url = "https://www.91kanju.com/vod-play/541-
    2-1.html"
136     main(url)
```

备注: 不是所有的视频网站都是这样的. 例如, bilibili, 爱奇艺等, 必须得单独想其他办法. 原因我就不讲了. 稍显复杂.