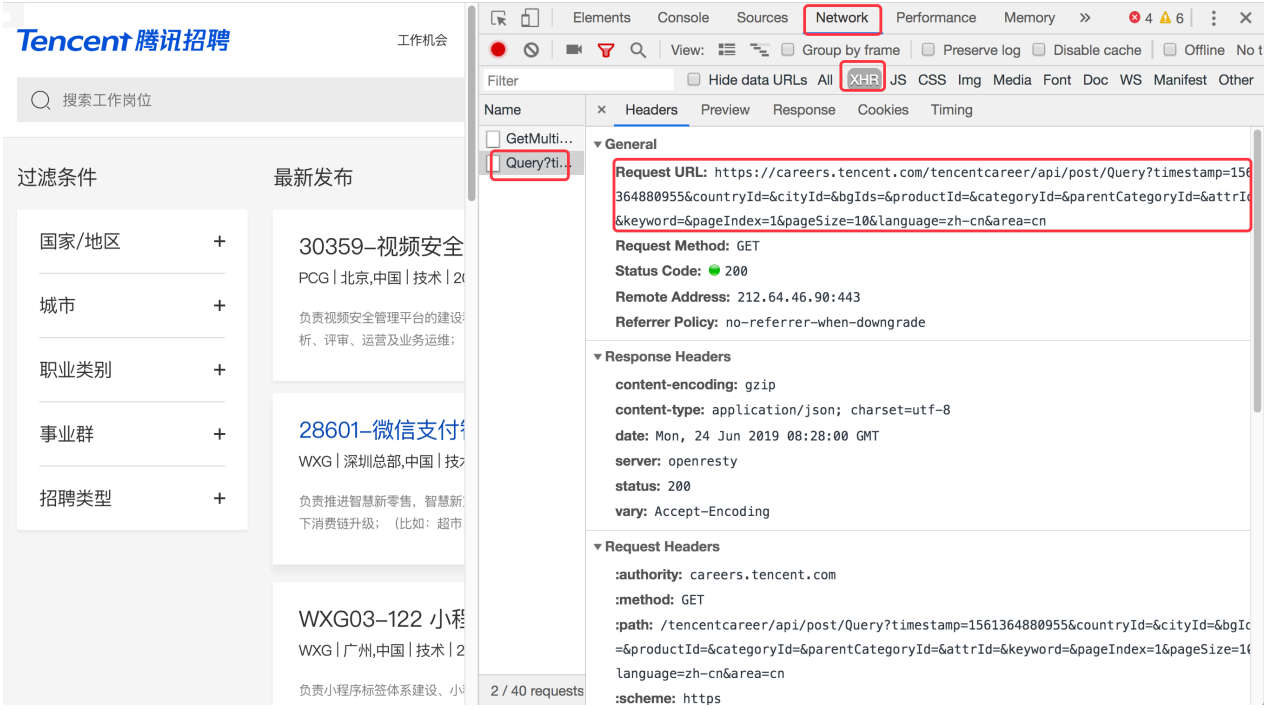
案例1: 腾讯招聘

1> 页面分析

抓取目标[腾讯招聘首页]

> 通过自行探索发现：下一页url为空
>
> 猜测：网页是否使用ajax发送数据?

还是[原来的页面]，打开谷歌浏览器开发者工具(按F12)，选择`Network`菜单项，指定`XHR`即只查看ajax请求数据：



- 去掉多余参数，可得请求url为

```
https://careers.tencent.com/tencentcareer/api/post/Query?
pageIndex=1&pageSize=10
```

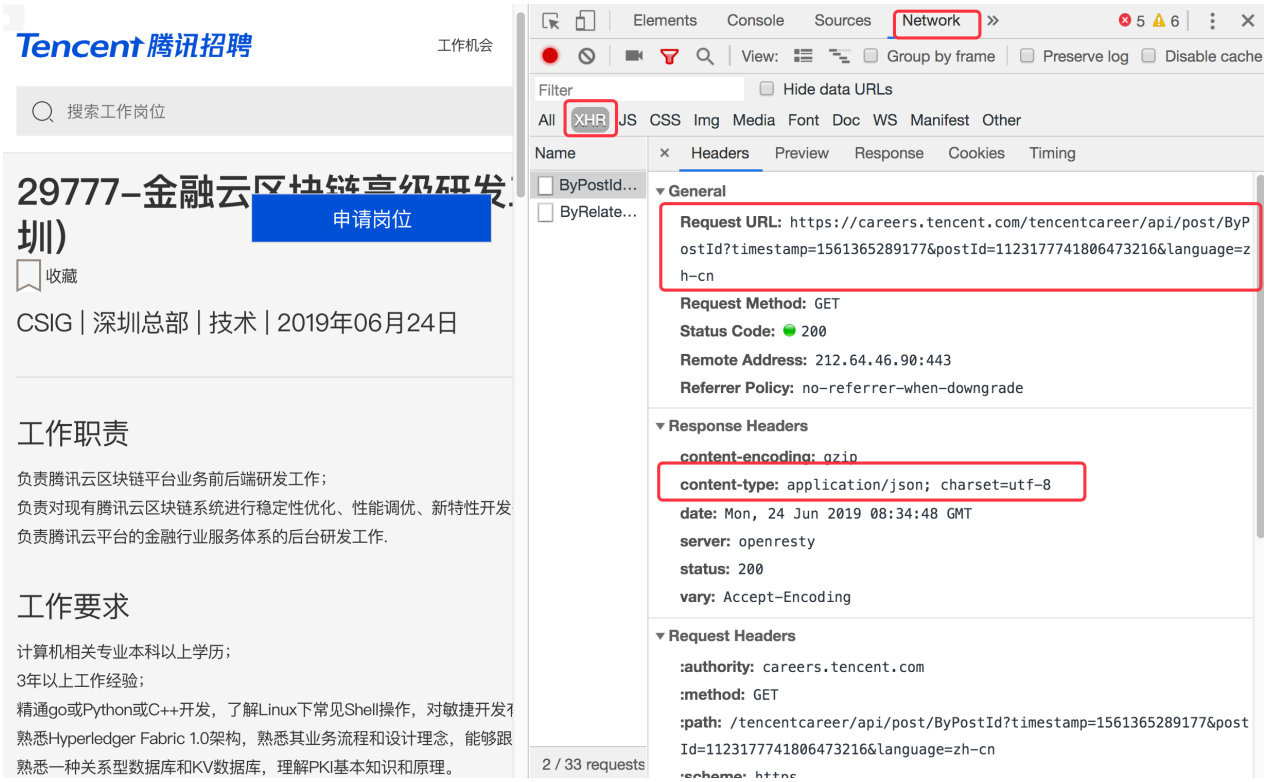
- 再次请求，查看json数据结构

```
{
  Code: 200,
- Data: {
    Count: 4156,
  - Posts: [
    - {
        Id: 0,
        PostId: "1143072200434126848",
        RecruitPostId: 51255,
        RecruitPostName: "22989-数据分析师(BI)",
        CountryName: "中国",
        LocationName: "深圳",
        BGName: "CSIG",
        ProductName: "",
        CategoryName: "技术",
        Responsibility: "1. 建立腾讯云业务数据分析体系，为业务管理决策与运营提供高质量数据服务。 2. 负责对应业务的数据梳理与数据接入，并设计数仓模型。 3. 负责对应业务的数据监控，分析业务指标变化趋势，并做出解释。 4. 结合各行业、业务运营现状与专项需求，提出优化改进措施并落地。 5.建立腾讯云数据中台。",
        LastUpdateTime: "2019年06月24日",
        PostURL: "http://careers.tencent.com/jobdesc.html?postId=0",
        SourceID: 1,
        IsCollect: false,
        IsValid: false
      },
    - {
        Id: 0,
        PostId: "1123177741806473216",
        RecruitPostId: 49097,
        RecruitPostName: "29777-金融云区块链高级研发工程师（深圳）",
        CountryName: "中国",
        LocationName: "深圳总部",
        BGName: "CSIG",
        ProductName: "",
        CategoryName: "技术",
        Responsibility: "负责腾讯云区块链平台业务前后端研发工作； 负责对现有腾讯云区块链系统进行稳定性优化、性能调优、新特性开发等工作；负责腾讯云平台的金融行业服务体系的后台研发工作.",
        LastUpdateTime: "2019年06月24日",
        PostURL: "http://careers.tencent.com/jobdesc.html?postId=1123177741806473216",
        SourceID: 1,
        IsCollect: false,
        IsValid: false
      },
```


Data.Posts[0].Responsibility

- 详情页

> 探索发现详情页数据也是使用ajax发送json数据



2> 新建项目及数据需求

- 新建爬虫项目

```
$ scrapy startproject Tencent
```

- 由页面分析可得能获取到的有效字段，编辑`item.py`文件

```
import scrapy


class TencentItem(scrapy.Item):
    # define the fields for your item here like:
    name = scrapy.Field()           # 岗位名称
    post_id = scrapy.Field()        # 岗位编号
    publish_date = scrapy.Field()   # 发布时间
    category = scrapy.Field()       # 工作分类
    duty = scrapy.Field()           # 岗位职责
    country = scrapy.Field()        # 国家
    location = scrapy.Field()       # 工作地点
    require = scrapy.Field()        # 工作要求
```

3> 新建爬虫并实现数据解析

- 新建爬虫

```
$ cd Tencent
$ scrapy genspider tencent 'tencent.com'
```

- 修改allowed_domains、start_urls

```
allowed_domains = ['careers.tencent.com']
start_urls =
[f"https://careers.tencent.com/tencentcareer/api/post/Query?
pageIndex={i}&pageSize=200" for i
              in range(1, 22)]
```

- 列表页数据解析

```
    def parse(self, response):
        data_json = json.loads(response.text)
        for data in data_json["Data"]["Posts"]:
            item = TencentItem()
            item["name"] = data["RecruitPostName"]
            item["post_id"] = data["PostId"]
            item["publish_date"] = data["LastUpdateTime"]
            item["category"] = data["CategoryName"]
            item["duty"] = data["Responsibility"]
            item["country"] = data["CountryName"]
            item["location"] = data["LocationName"]
```

- 发起详情页请求

```
    detail_url = "https://careers.tencent.com/tencentcareer/api/post/ByPostId?postId={}"

    def parse(self, response):
        ......
        yield scrapy.Request(
                self.detail_url.format(item["post_id"]),
                callback=self.parse_detail,
                meta=dict(item=item)
            )
```

- 详情页数据解析

```
   def parse_detail(self, response):
        """详情页解析"""
        item = response.meta["item"]
        data_json = json.loads(response.text)
        item["require"] = data_json["Data"]["Requirement"]
        yield item
```

4> 数据持久化及配置文件

- 把数据保存到mongoDB，编辑piplines.py

```
from datetime import datetime

from pymongo import MongoClient
from scrapy.conf import settings


class TencentPipeline(object):

    def __init__(self):
        # 读取配置参数
        host = settings['MONGO_HOST']
        port = settings['MONGO_PORT']
        db_name = settings['MONGO_DBNAME']
        col_name = settings['MONGO_COLNAME']

        now_str = datetime.now().strftime("%m_%d_%H_%M")
        col_name += "_" + now_str

        # 连接mongodb数据库
        self.handle = MongoClient(host, port)
        # 选择数据库
        self.db = self.handle[db_name]
        # 选择集合
        self.col = self.db[col_name]

    def process_item(self, item, spider):
        # 1. 使用post_id做主键
        data = dict(item)  # 将item实例转化为字典
        # 2. 查询库中数据是否已存在，不存在才存库
        res = self.col.find_one({"_id": data["post_id"]})
        if not res:
            data["_id"] = data.pop("post_id")
            self.col.insert(data)  # 写入数据库
        return item
```

```
    def close_spider(self, spider):
        # 关闭数据库链接
        self.handle.close()
```

- 配置文件settings.py

```
LOG_LEVEL = "WARNING"

MONGO_HOST = '127.0.0.1'
MONGO_PORT = 27017
MONGO_DBNAME = 'spider'
MONGO_COLNAME = 'tencent'

USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106
Safari/537.36'

......

ITEM_PIPELINES = {
   'Tencent.pipelines.TencentPipeline': 300,
}
```