HW02a Report

1. 爬虫实现

目标网站:网易云音乐 (https://music.163.com)

通过观察页面，发现规律。有以下难点：

- 网页具体内容是在标签 `<iframe>` 动态生成的，直接爬取无法提取实质内容。
- 音乐文件的播放外链是经过加密的，检测请求和响应无法获取真实的播放外链。

解决方法

- 使用Selenium框架下的webdriver模拟浏览器，获取动态部分内容。此处使用的是chrome的驱动。

```
#配置webdriver
chrome_options=Options()
chrome_options.add_argument('-headless')
chromedriver="./chromedriver.exe"
driver = webdriver.Chrome(chromedriver,chrome_options=chrome_options)

#获取隐藏html文件
def getHttp(url):
        driver.get(url)
        iframe = driver.find_elements_by_tag_name('iframe')[0]
        driver.switch_to.frame(iframe)   # 最重要的一步
        soup = BeautifulSoup(driver.page_source, "html.parser")
        return soup
```

- 通过查阅相关资料，获取实际的播放外链
- 爬取逻辑:

1. 观察url 获取歌单id
2. 在某歌单id下爬取每首歌的id,歌曲id位于某个 `<a>` 的content之中，这个 `<a>` 的路径为 `'td > div > div > div > span > a'`

```
data=soup.select('td > div > div > div > span > a')
```

3. 根据歌曲id，拼出某歌曲页面:

```
#歌曲主页面
url='https://music.163.com/#/song?id='+songid
#歌曲mp3播放url
url='http://music.163.com/song/media/outer/url?id='+songid+'.mp3'
#歌词url
url='http://music.163.com/api/song/lyric?' + 'id=' + songid + '&lv=1&kv=1&tv=-1'
```

4. 在歌曲页面获取相关文字信息，以及封面图片的链接（该链接未加密）。这些信息都在 `<head><meta>` 下,并且用 ， 。分隔开，于是我们使用split()函数，并将相关信息存储到dict中，以便存储相关信息建立metadata

```
#获取曲名
keywords=soup.select('meta')[5]
keywords=keywords.get('content')
keywords=keywords.split(", ")
object["title"]=keywords[0]
#获取artist和专辑信息
descript=soup.select('meta')[6]
descript=descript.get('content')
descript=descript.split("。")
object["artist"]=descript[0][3:]
object["album"]=descript[1][5:]
#获取图片链接
img_url=soup.select('meta')[9]
img_url=img_url.get('content')
```

5. 每爬取一首歌，即建立一个对象，包含下载到本地的资源和对象基本描述,写入meta.json文件中，用 '\n' 分隔对象

```
        songs=getsongid('/playlist?id='+LISTID)
        for song in songs:
                id=song.get("href")[9:]
                song_object={}
                song_object["id"]=id
                song_object["category"]=CATEGORY
                getsong(song_object)
                getimg_descript(song_object)
                getlyrics(song_object)
                #写入索引
                with open(FOLDER+"/meta.json","a+") as code:
                        code.write(json.dumps(song_object))
                        code.write("\n")
```

2. 修改HW02中的网站架构

1. 建立分区索引
   在资源总文件夹下建立索引，记录当前资源分区的文件夹。

```
{
        "media_src":[
                "chinese",
                "japanese",
                "russian",
                "minions",
                "ringtone"],
        "web_img":"image",
        "scripts":"scripts",
        "content":"content",
        "fonts":"fonts"
}
```

服务器读取索引文件，获取分区实际情况：

```
with open("./myapp/static/meta.json","r+",encoding="utf-8") as code:
        obj=json.load(code)
        fields=obj["media_src"]
```

在渲染页面时，使用循环，动态加载。以导航栏为例：

```
{% for field in fields %}
    <li><a href="{{url_for('field',name=field)}}">{{field}}</a></li>
{% endfor %}
```

3. 建立资源索引
   在分区文件夹下建立索引，记录当前资源信息,当某特定页面请求时，返回索引内容，以便页面加载资源。

```
def getmeta(path):
        list=[]
        with open(path,"r+",encoding="utf-8") as code:
                for line in code:
                        obj=json.loads(line)
                        list.append(obj)
        return list
```

5. .lrc文件解析
   这种文件被普遍用于显示歌词，
   基本格式为: [时间或者'ar'\'ti'\'al']<内容>\n
   根据格式分别提取时间和内容，在dict中记录下来。
   具体解析方式不加赘述。