本课时我们主要学习如何使用 Charles。

Charles 是一个网络抓包工具，我们可以用它来做 App 的抓包分析，得到 App 运行过程中发生的所有网络请求和响应内容，这就和 Web 端浏览器的开发者工具 Network 部分看到的结果一致。

Charles、Fiddler 等都是非常强大的 HTTP 抓包软件，功能基本类似，不过 Charles 的跨平台支持更好。所以我们选用 Charles 作为主要的移动端抓包工具，用于分析移动 App 的数据包，辅助完成 App 数据抓取工作。

本节目标

本节我们以电影示例 App 为例，通过 Charles 抓取 App 运行过程中的网络数据包，然后查看具体的 Request 和 Response 内容，以此来了解 Charles 的用法。

同时抓取到数据包之后，我们采用 Python 将请求进行改写，从而实现 App 数据的爬取。

准备工作

请确保已经正确安装 Charles 并开启了代理服务，另外准备一部 Android 手机，系统版本最好是在 7.0 以下。

> 如果系统版本在 7.0 及以上，可能出现 SSL Pining 的问题，可以参考第一课时的思路来解决。

然后手机连接 Wi-Fi，和 PC 处于同一个局域网下，另外将 Charles 代理和 Charles CA 证书设置好，同时需要开启 SSL 监听。

此过程的配置流程可以参见：https://cuiqingcai.com/5255.html。

最后手机上安装本节提供的 apk（apk 随课件一同领取），进行接下来的 Charles 抓包操作。

原理

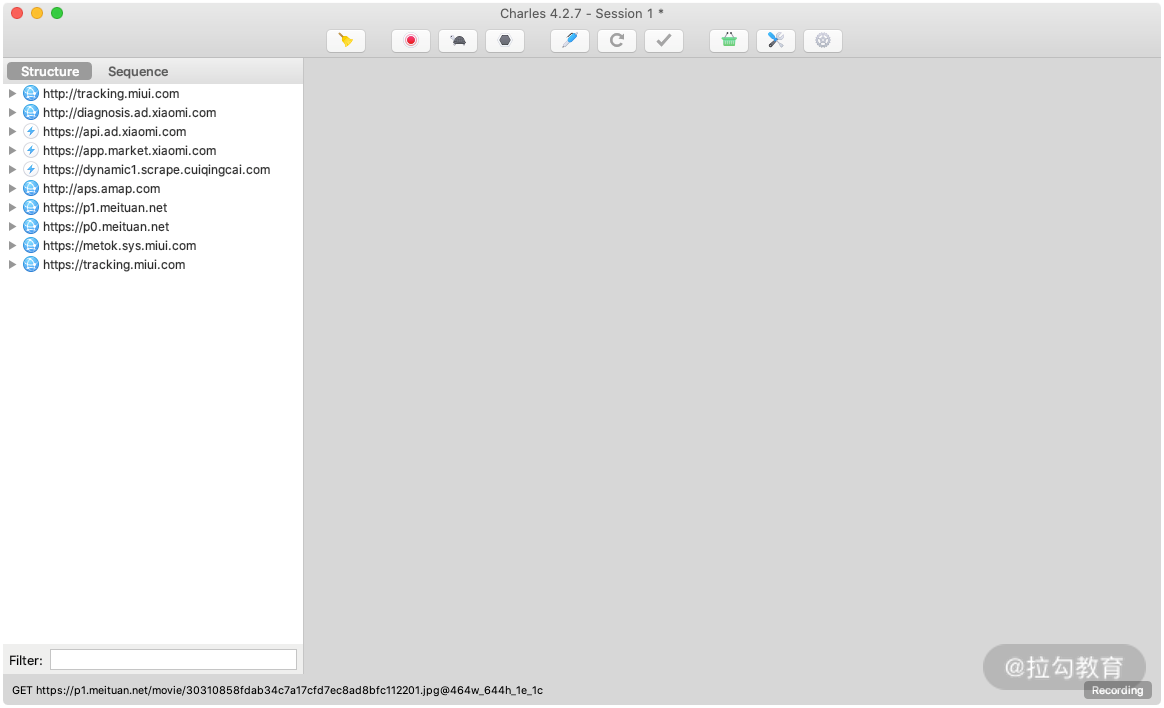
首先将 Charles 运行在自己的 PC 上，Charles 运行的时候会在 PC 的 8888 端口开启一个代理服务，这个服务实际上是一个 HTTP/HTTPS 的代理。

确保手机和 PC 在同一个局域网内，我们可以使用手机模拟器通过虚拟网络连接，也可以使用手机真机和 PC 通过无线网络连接。

设置手机代理为 Charles 的代理地址，这样手机访问互联网的数据包就会流经 Charles，Charles 再转发这些数据包到真实的服务器，服务器返回的数据包再由 Charles 转发回手机，Charles 就起到中间人的作用，所有流量包都可以捕捉到，因此所有 HTTP 请求和响应都可以捕获到。同时 Charles 还有权力对请求和响应进行修改。

抓包

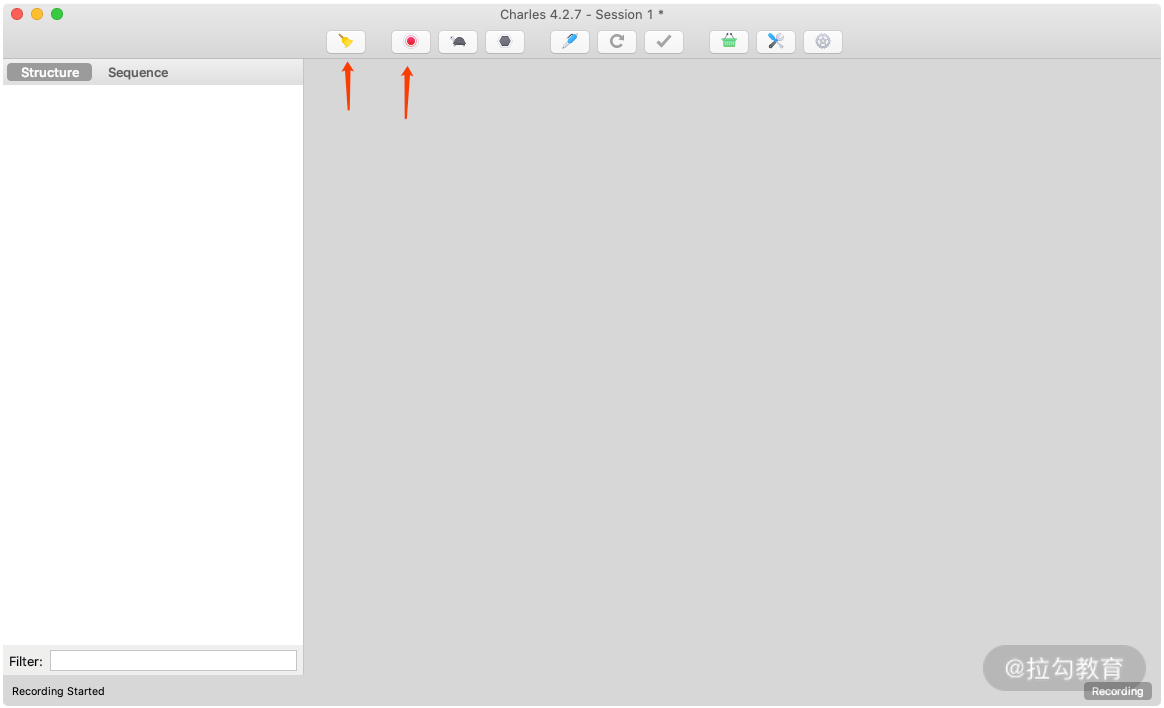
好，我们先打开 Charles，初始状态下 Charles 的运行界面如图所示。



Charles 会一直监听 PC 和手机发生的网络数据包，捕获到的数据包就会显示在左侧，随着时间的推移，捕获的数据包越来越多，左侧列表的内容也会越来越多。

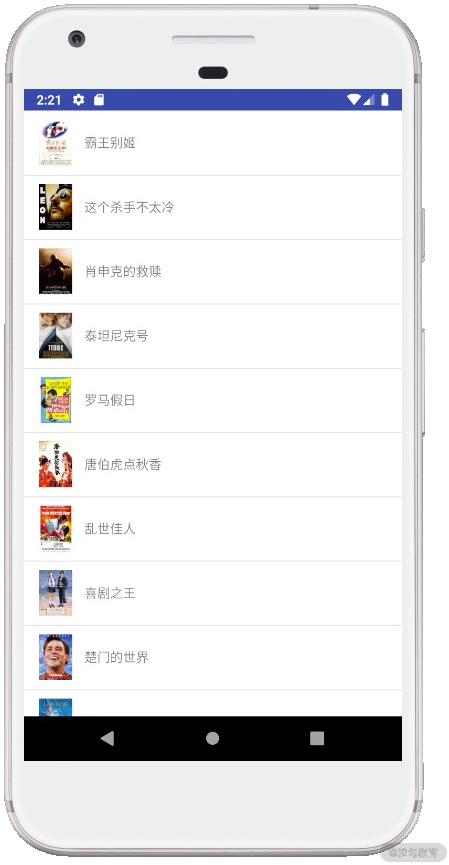
可以看到，图中左侧显示了 Charles 抓取到的请求站点，我们点击任意一个条目便可以查看对应请求的详细信息，其中包括 Request、Response 等内容。

接下来清空 Charles 的抓取结果，点击左侧的扫帚按钮即可清空当前捕获到的所有请求。然后点击第二个监听按钮，确保监听按钮是打开的，这表示 Charles 正在监听 App 的网络数据流，如图所示。

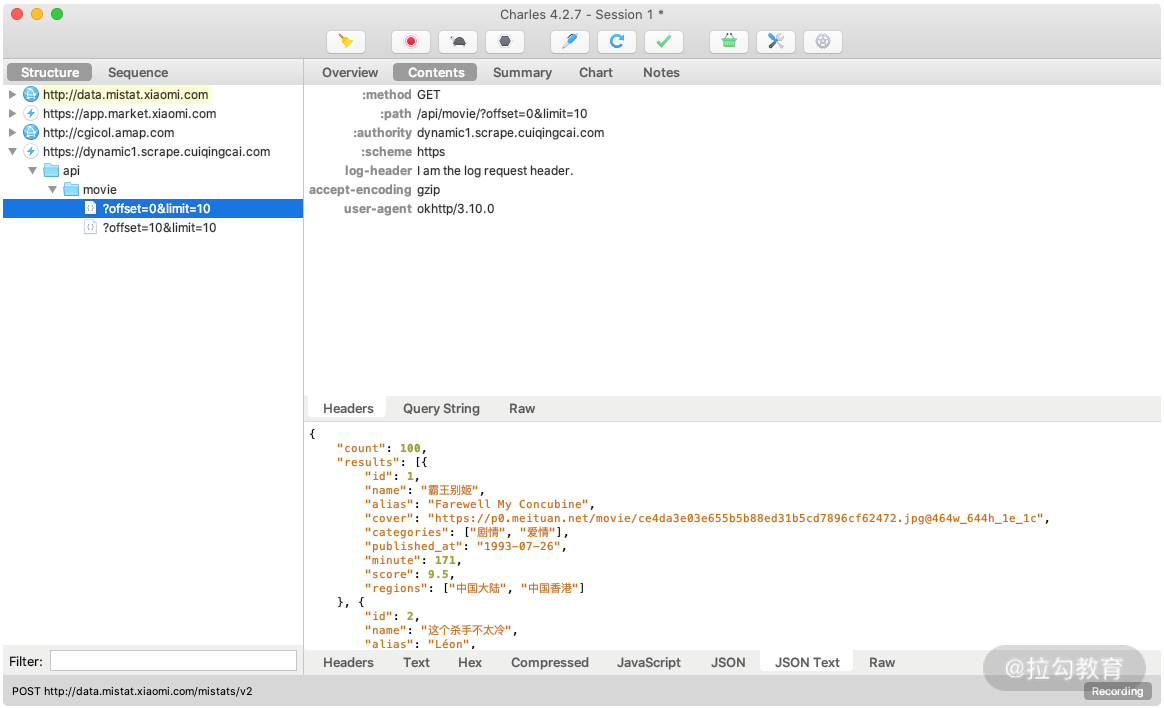


这时打开 App，注意一定要提前设置好 Charles 的代理并配置好 CA 证书，否则没有效果。

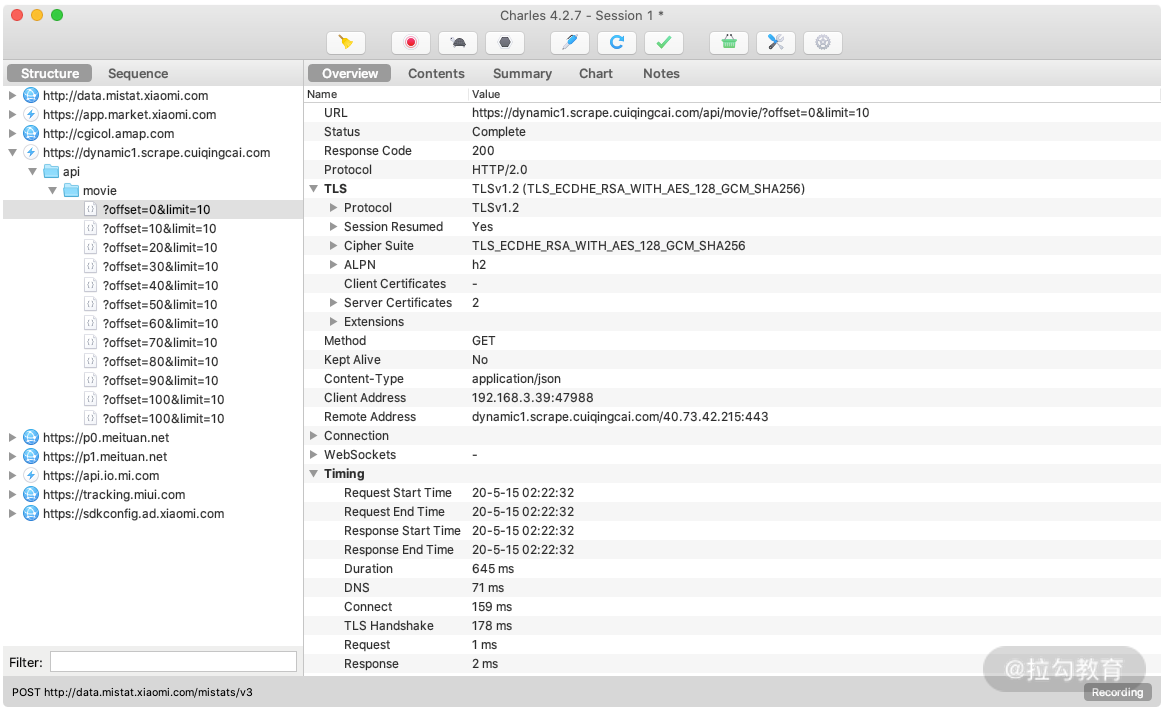
打开 App 之后我们就可以看到类似如下的页面。



这时候我们就可以发现 Charles 里面已经抓取到了对应的数据包，出现了类似如图所示的结果。

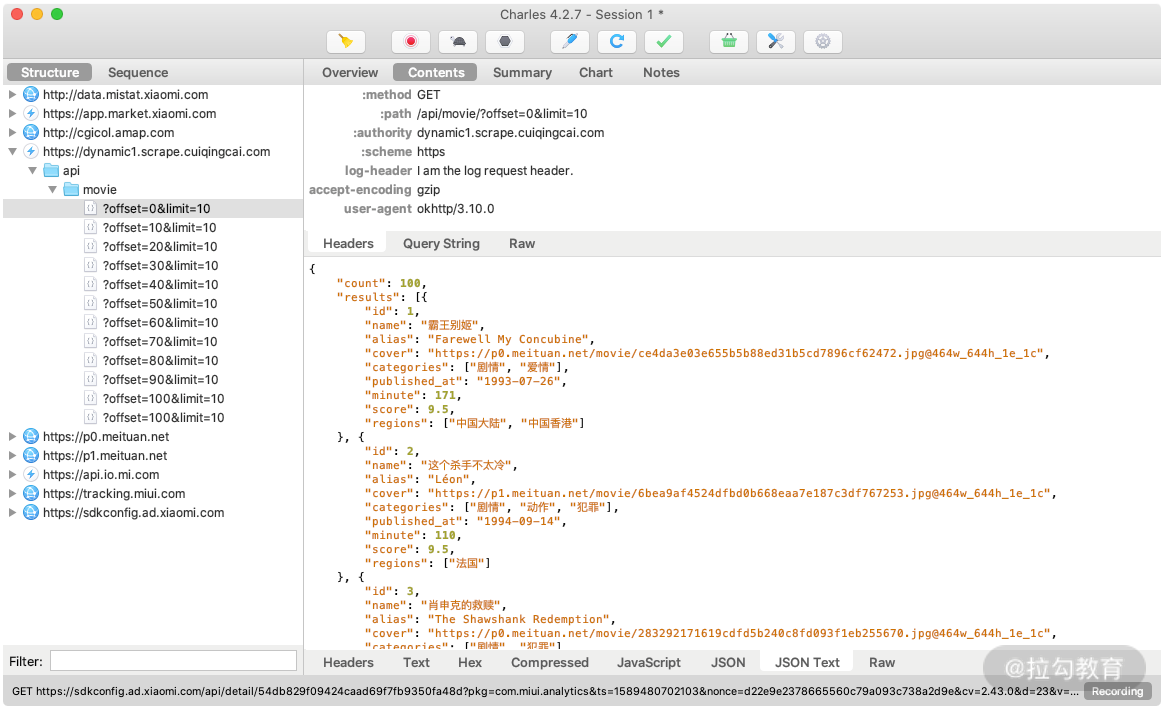


我们在 App 里不断上拉，可以看到 Charles 捕获到这个过程中 App 内发生的所有网络请求，如图所示。



左侧列表中会出现一个 dynamic1.scrape.cuiqingcai.com 的链接，而且在 App 上拉过程它在不停闪动，这就是当前 App 发出的获取数据的请求被 Charles 捕获到了。

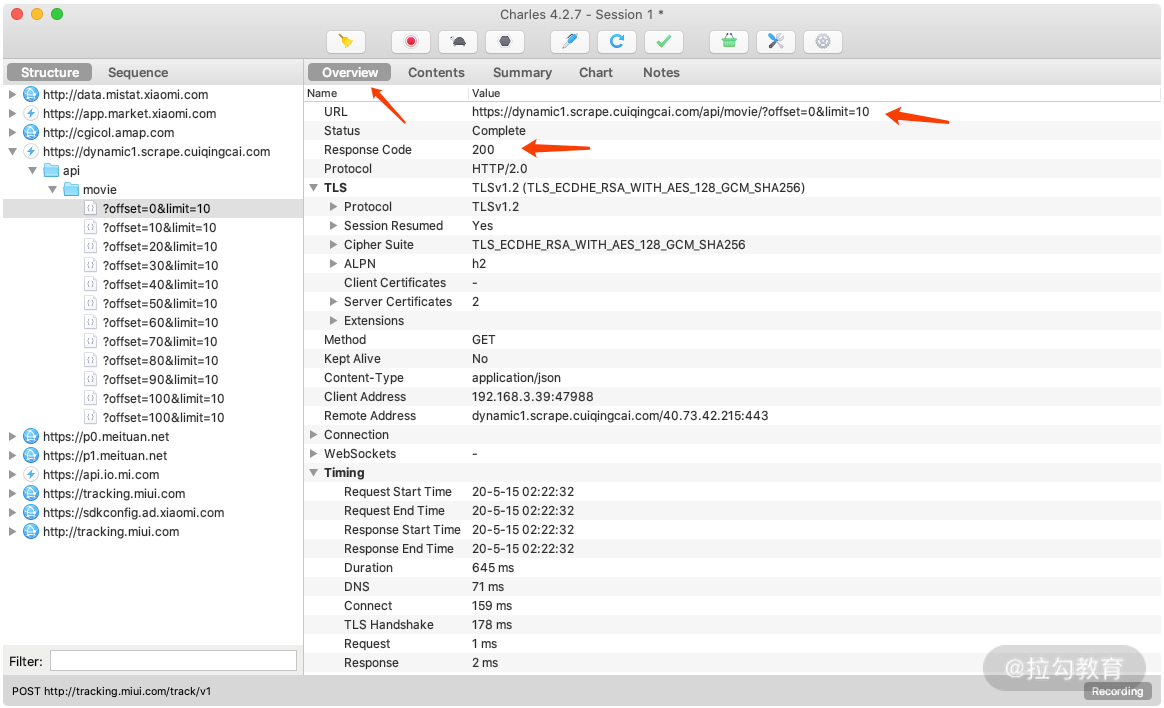
为了验证其正确性，我们点击查看其中一个条目的详情信息。切换到 Contents 选项卡，这时我们发现一些 JSON 数据，核对一下结果，结果有 results 字段，每一个条目的 name 字段就是电影的信息，这与 App 里面呈现的内容是完全一致的，如图所示。



这时可以确定，此请求对应的接口就是获取电影数据的接口。这样我们就成功捕获到了在上拉刷新的过程中发生的请求和响应内容。

分析

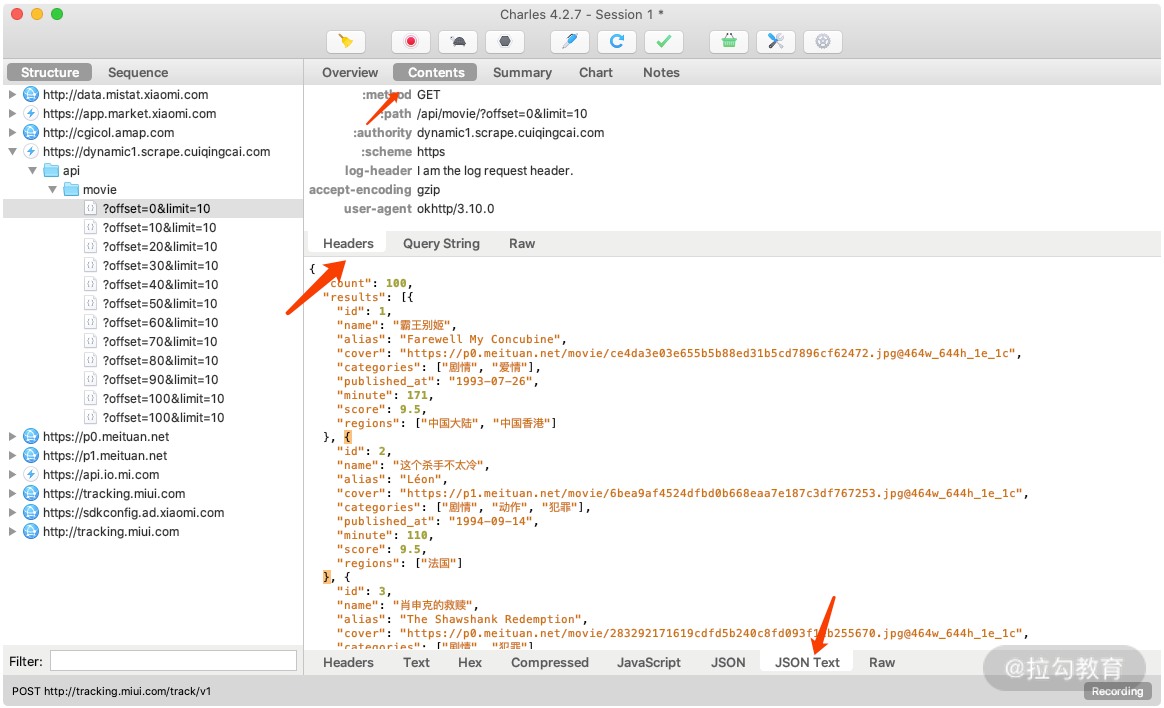
现在分析一下这个请求和响应的详细信息。首先可以回到 Overview 选项卡，上方显示了请求的接口 URL，接着是响应状态 Status Code、请求方式 Method 等，如图所示。



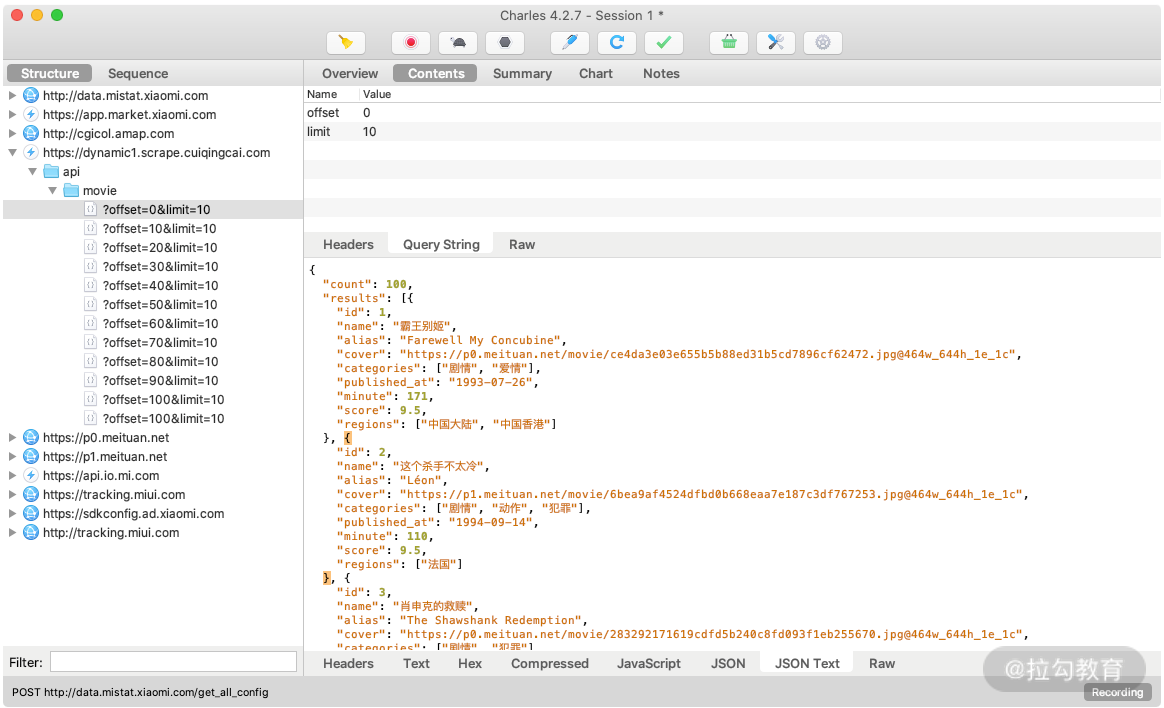
这个结果和原本在 Web 端用浏览器开发者工具内捕获到的结果形式是类似的。

接下来点击 Contents 选项卡，查看该请求和响应的详情信息。

上半部分显示的是 Request 的信息，下半部分显示的是 Response 的信息。比如针对 Reqeust，我们切换到 Headers 选项卡即可看到该 Request 的 Headers 信息，针对 Response，我们切换到 JSON Text 选项卡即可看到该 Response 的 Body 信息，并且该内容已经被格式化，如图所示。



由于这个请求是 GET 请求，所以我们还需要关心的就是 GET 的参数信息，切换到 Query String 选项卡即可查看，如图所示。

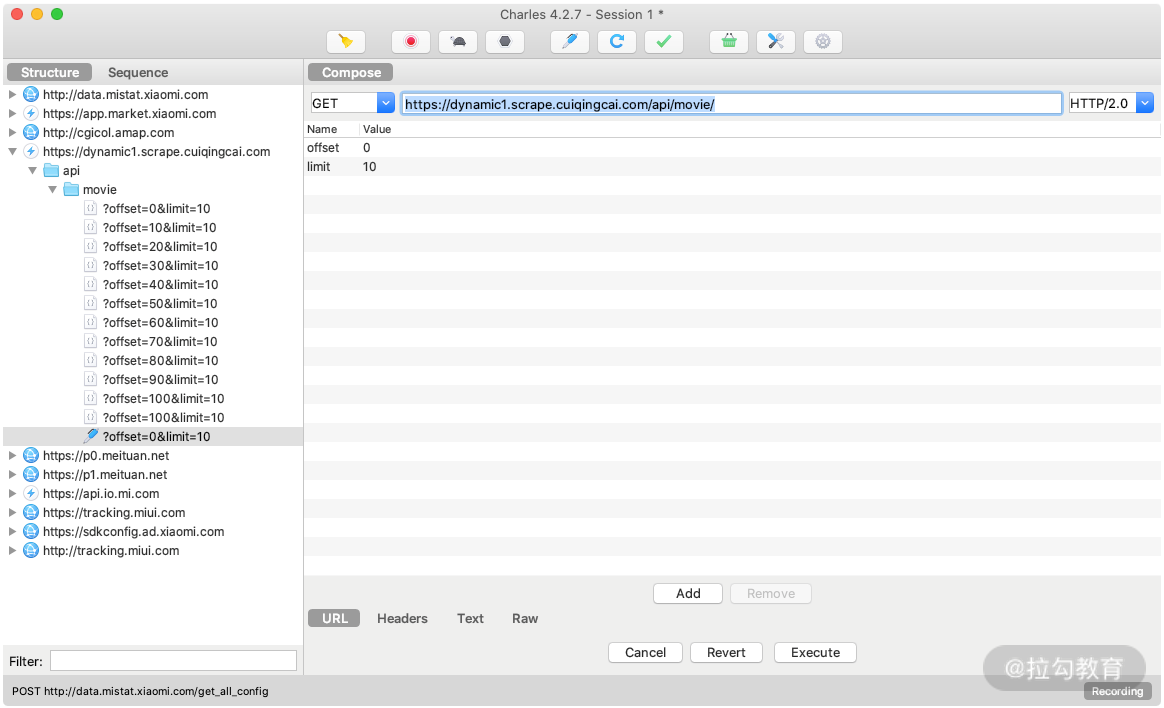


这样我们就成功抓取到了 App 中的电影数据接口的请求和响应，并且可以查看 Response 返回的 JSON 数据。

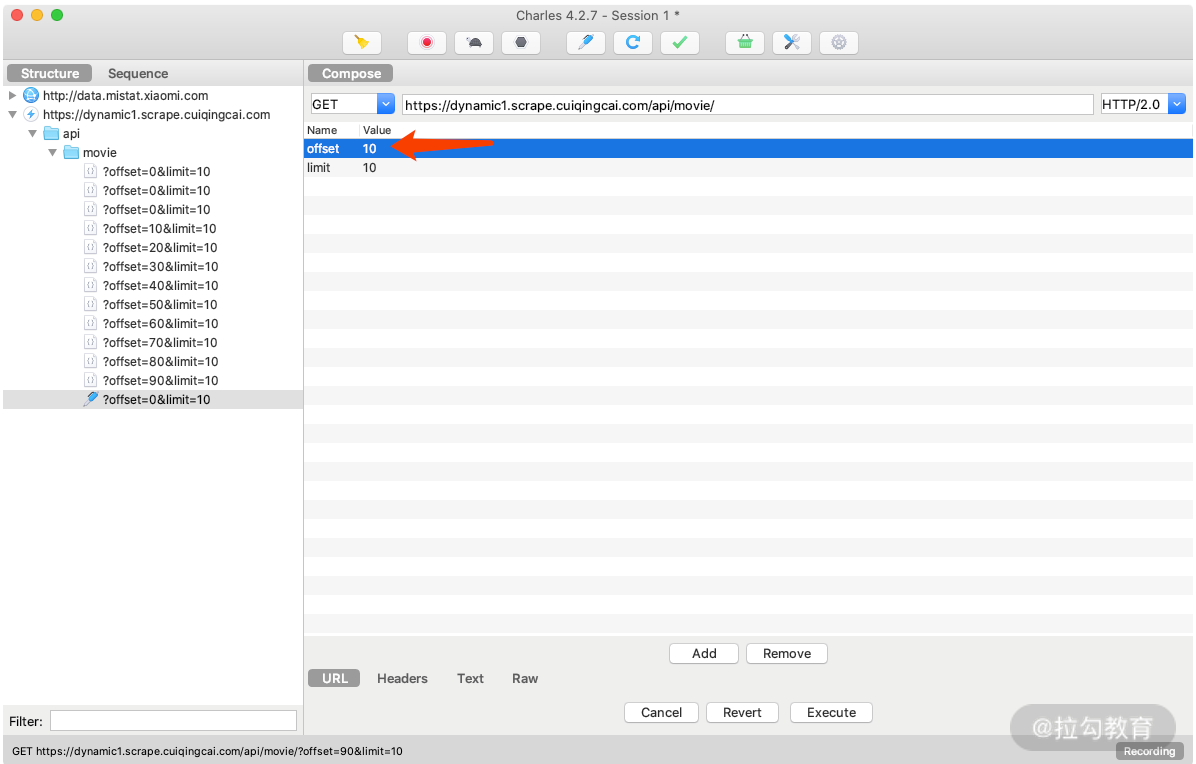
至于其他 App，我们同样可以使用这样的方式来分析。如果我们可以直接分析得到请求的 URL 和参数的规律，直接用程序模拟即可批量抓取。

重发

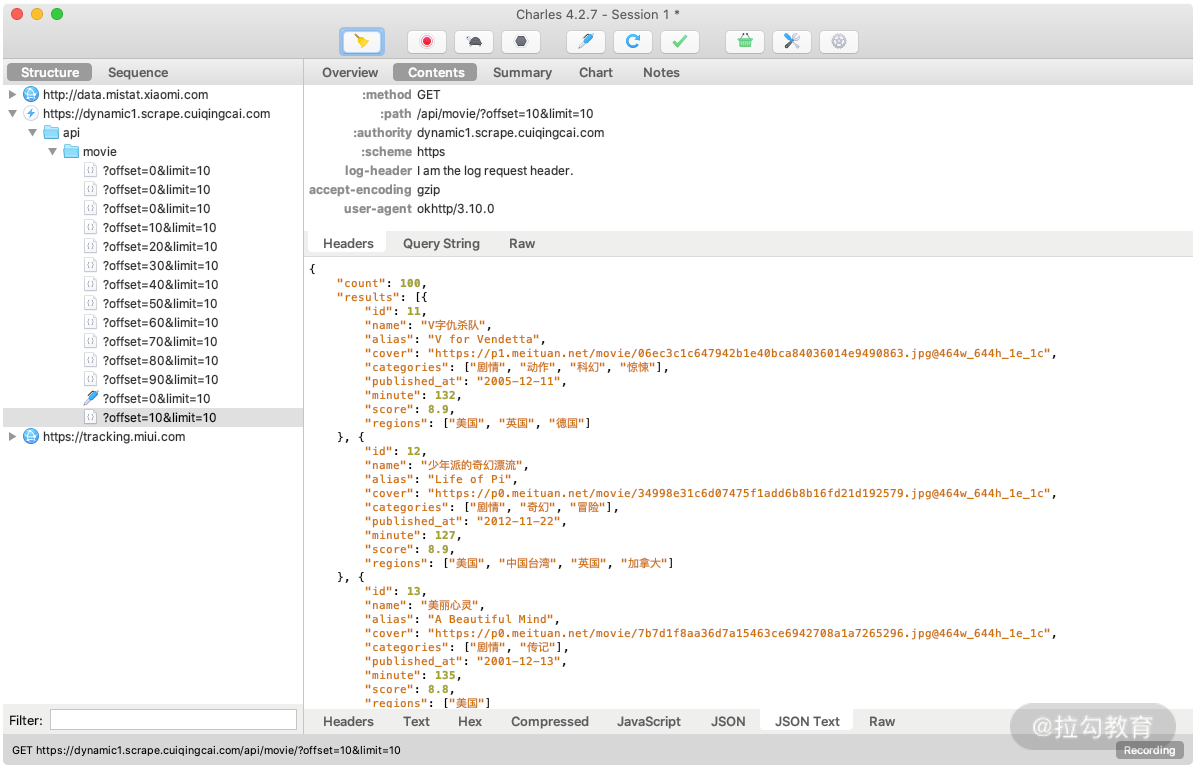
Charles 还有一个强大功能，它可以将捕获到的请求加以修改并发送修改后的请求。点击上方的修改按钮，左侧列表就多了一个以编辑图标为开头的链接，这就代表此链接对应的请求正在被我们修改，如图所示。



我们可以将参数中的某个字段修改下，比如这里将 offset 字段由 0 修改为 10。这时我们已经对原来请求携带的 Query 参数做了修改，然后点击下方的 Execute 按钮即可执行修改后的请求，如图所示。



可以发现左侧列表再次出现了接口的请求结果，内容变成了第 11~20 条内容，如图所示。



有了这个功能，我们就可以方便地使用 Charles 来做调试，可以通过修改参数、接口等来测试不同请求的响应状态，就可以知道哪些参数是必要的哪些是不必要的，以及参数分别有什么规律，最后得到一个最简单的接口和参数形式以供程序模拟调用使用。

模拟

现在我们已经成功完成了抓包操作了，所有的请求一目了然，请求的 URL 就是 https://dynamic1.scrape.cuiqingcai.com/api/movie/，后面跟了两个 GET 请求参数。经过观察，可以很轻松地发现 offset 就是偏移量，limit 就是一次请求要返回的结果数量。比如 offset 为 20，limit 为 10，就代表获取第 21~30 条数据。另外我们通过观察发现一共就是 100 条数据，offset 从 0 到 90 遍历即可。

接下来我们用 Python 简单实现一下模拟请求即可，这里写法一些从简了，代码如下：

```
import requests

BASE_URL = 'https://dynamic1.scrape.cuiqingcai.com/api/movie?offset={offset}&limit=10'
for i in range(0, 10):
    offset = i * 10
    url = BASE_URL.format(offset=offset)
    data = requests.get(url).json()
    print('data', data)
```

运行结果如下：

```
data {'count': 100, 'results': [{'id': 1, 'name': '霸王别姬', 'alias': 'Farewell My Concubine', 'cover': 'https://p0.meituan.net/movie/ce4da3e03e655b5b88ed31b5cd7896cf62472.jpg@464w_644h

data {'count': 100, 'results': [{'id': 11, 'name': 'V字仇杀队', 'alias': 'V for Vendetta', 'cover': 'https://p1.meituan.net/movie/06ec3c1c647942b1e40bca84036014e9490863.jpg@464w_644h_1e_

data {'count': 100, 'results': [{'id': 21, 'name': '黄金三镖客', 'alias': 'Il buono, il brutto, il cattivo.', 'cover': ...
```

可以看到每个请求都被我们轻松模拟实现了，数据也被爬取下来了。

由于这个 App 的接口没有任何加密，所以仅仅靠抓包完之后观察规律我们就能轻松完成 App 接口的模拟爬取。

结语

以上内容便是通过 Charles 抓包分析 App 请求的过程。通过 Charles，我们成功抓取 App 中流经的网络数据包，捕获原始的数据，还可以修改原始请求和重新发起修改后的请求进行接口测试。

知道了请求和响应的具体信息，如果我们可以分析得到请求的 URL 和参数的规律，直接用程序模拟即可批量抓取，这当然最好不过了。

但是随着技术的发展，App 接口往往会带有密钥或者无法抓包，后面我们会继续讲解此类情形的处理操作。