关于HTTP协议，本文以及后续，如无特殊说明，我们的讨论是基于目前主流的HTTP/1.1版本。

还记得我们的传输层、网络层吗？他们都有自己的头部信息来标识。HTTP 协议也是与 TCP/UDP 类似，同样也需要在实际传输的数据前附加一些头数据，不过与 TCP/UDP 不同的是，**HTTP/1.1版本中头部信息是一个“纯文本”的协议，所以头数据都是 ASCII 码的文本，可以很容易地用肉眼阅读，不用借助程序解析也能够看懂**。

用于HTTP的协议交互的信息被称为HTTP报文。请求端的HTTP报文叫做请求报文，响应端的叫做响应报文。

由于HTTP报文字段也比较多，本篇文章先整体看下报文结构。

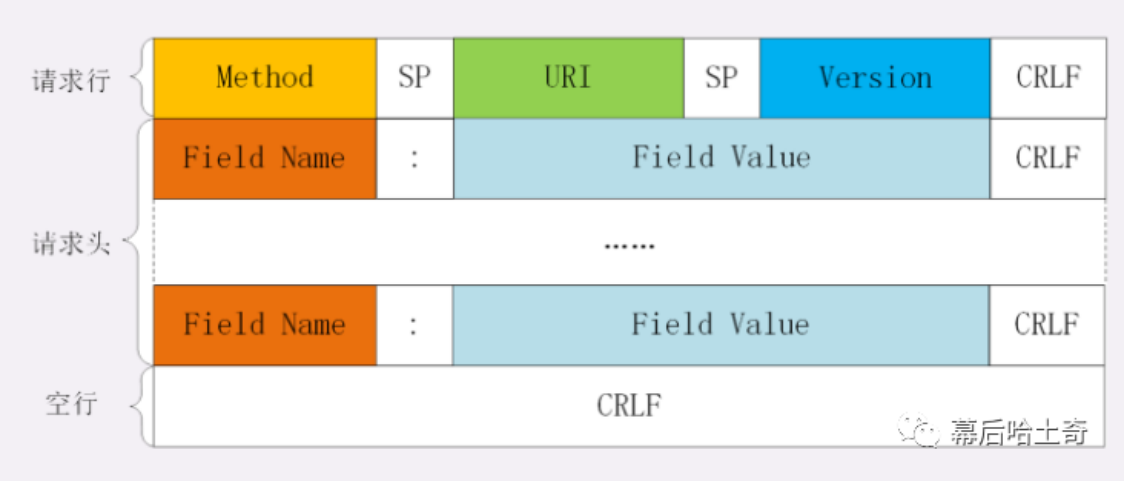
一、请求报文

HTTP的这两种报文都由四部分组成：

- **起始行**：包含【请求的方法如GET、POST】，【请求URI】和【HTTP版本】，中间用空格分隔
- **头部**：包含请求的各种条件和属性，使用key:value的形式展示
- **空行**：它的作用是通过一个空行，告诉服务器请求头部到此为止
- **实体**：实际传输的数据，不一定是文本，可以是图片视频等二进制文件。若方法字段是GET，则此项为空，没有数据；若方法字段是POST,则通常来说此处放置的就是要提交的数据

在请求报文中，我们往往将起始行和头部合称为请求头，即Header；我们将实体常叫做请求体，即Body。

其中请求头整体结构为：



HTTP协议规定报文必须有 header ，可以没有 body ，比如常用的 GET 请求。
且 header 和 body 中间必须有个空行以进行区分。

一个GET请求一般实际上长这个样子：

```
GET / HTTP/1.1
Host: 127.0.0.1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image
/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
```

嘉后哈士奇

在这个浏览器发出的请求报文里，第一行“GET / HTTP/1.1”就是请求行，其中 GET 就是请求方法， / 就是请求的 URI ，HTTP/1.1 就是HTTP协议及版本；

而后面的“Host”、“Connection”等等都属于 header 部分；

报文的最后是一个空白行结束，且没有body。

二、响应报文

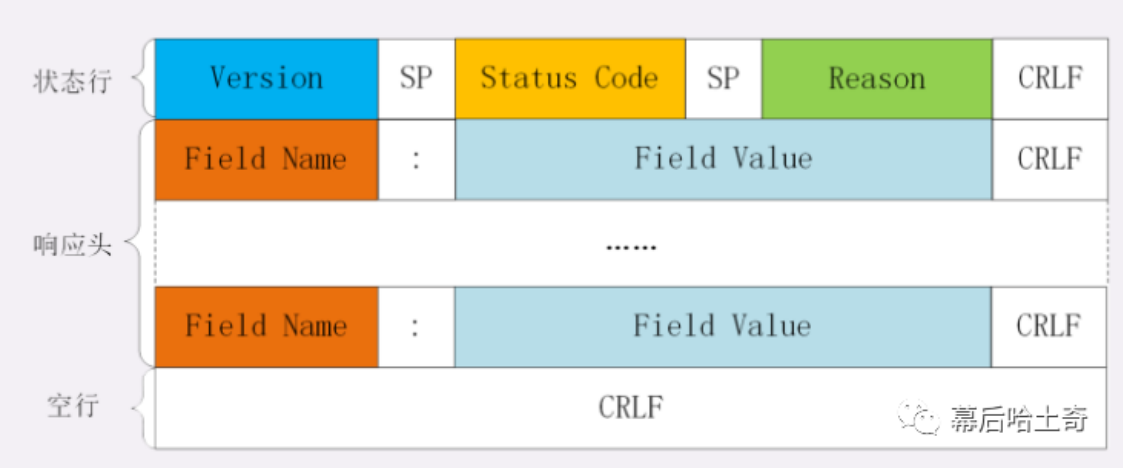
对应地，HTTP响应报文也由四部分组成：

- **起始行**：响应行一般由【协议版本】、【状态码如200】及其【描述】组成 比如 HTTP/1.1 200 OK

- **头部**：响应头用于描述服务器的基本信息，以及数据的描述，服务器通过这些数据的描述信息，可以通知客户端如何处理等一会儿它回送的数据。
- **空行**：它的作用是通过一个空行，告诉客户端请求头部到此为止。
- **响应实体**：响应体就是响应的消息体，如果是纯数据就是返回纯数据，如果请求的是HTML页面，那么返回的就是HTML代码，如果是JS就是JS代码，如此之类。

在响应报文中，我们往往将起始行和头部合称为响应头；我们将实体常叫做响应体。

响应头整体结构为：



可见，请求头和响应头的结构是基本一样的，唯一的区别是起始行。

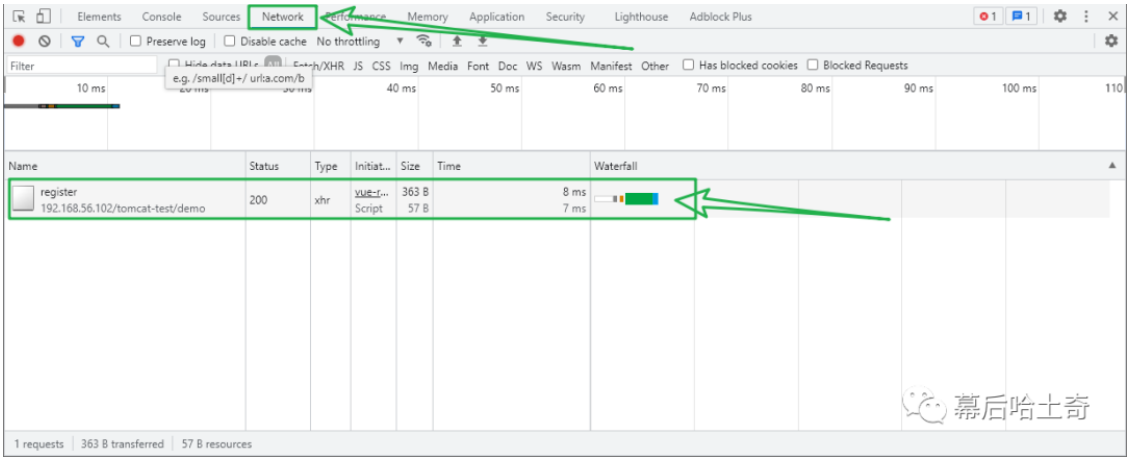
HTTP 头字段非常灵活，不仅可以使用标准里的 Host、Connection 等已有头，也可以任意添加自定义头，这就给 HTTP 协议带来了无限的扩展可能。

三、浏览器上查看

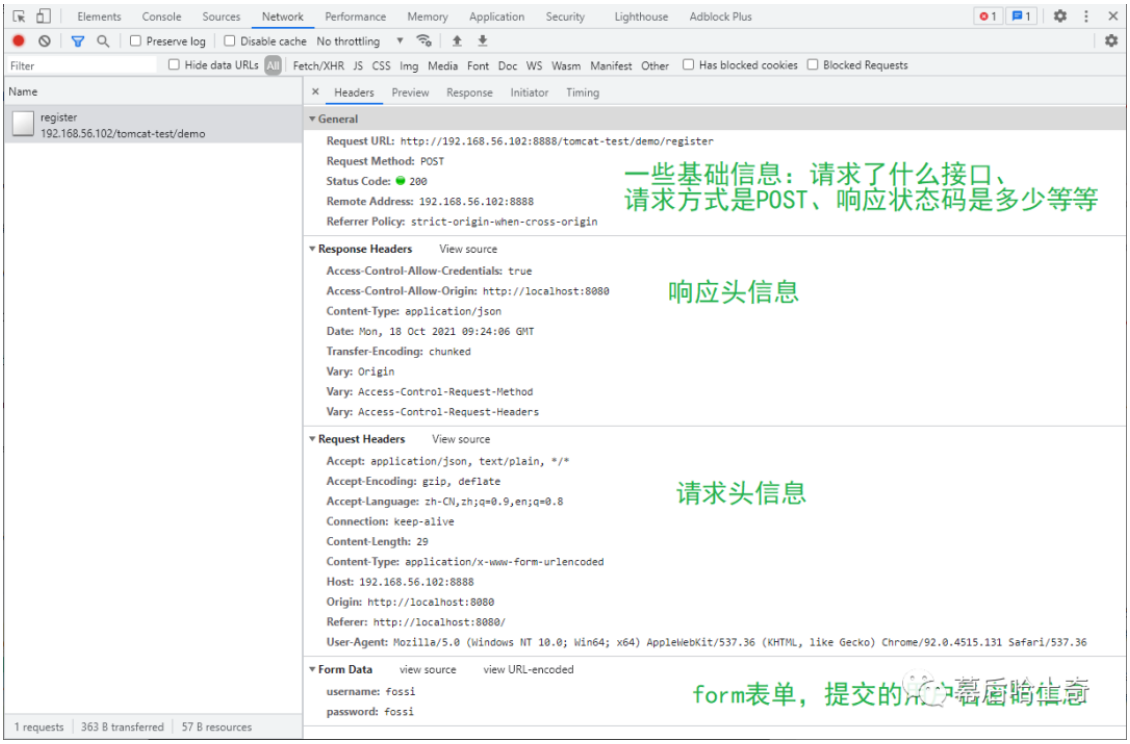
我们在浏览器上即可进行HTTP报文的查看，这个对我们平时排查问题很重要，操作十分简单，可见当代浏览器的强大。

启动之前我们搭建的前后端分离项目，进行用户注册操作。

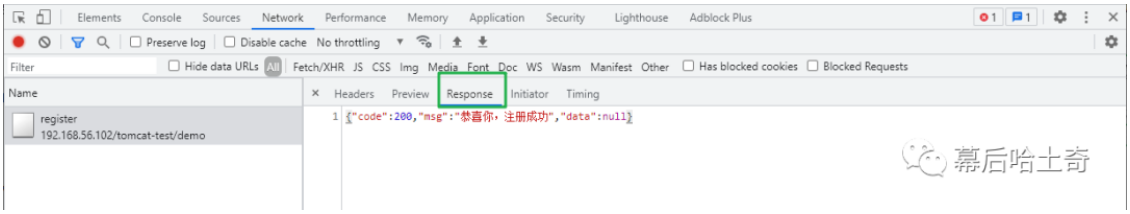
点击F12打开“开发者工具”，我们在开发者工具中可以捕获到此用户注册请求。



鼠标点击左边这个捕获到的报文，即可看到详细的请求和响应信息。



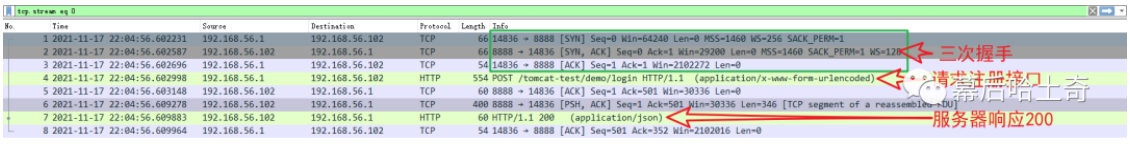
服务器返回的信息也可以直接看到：



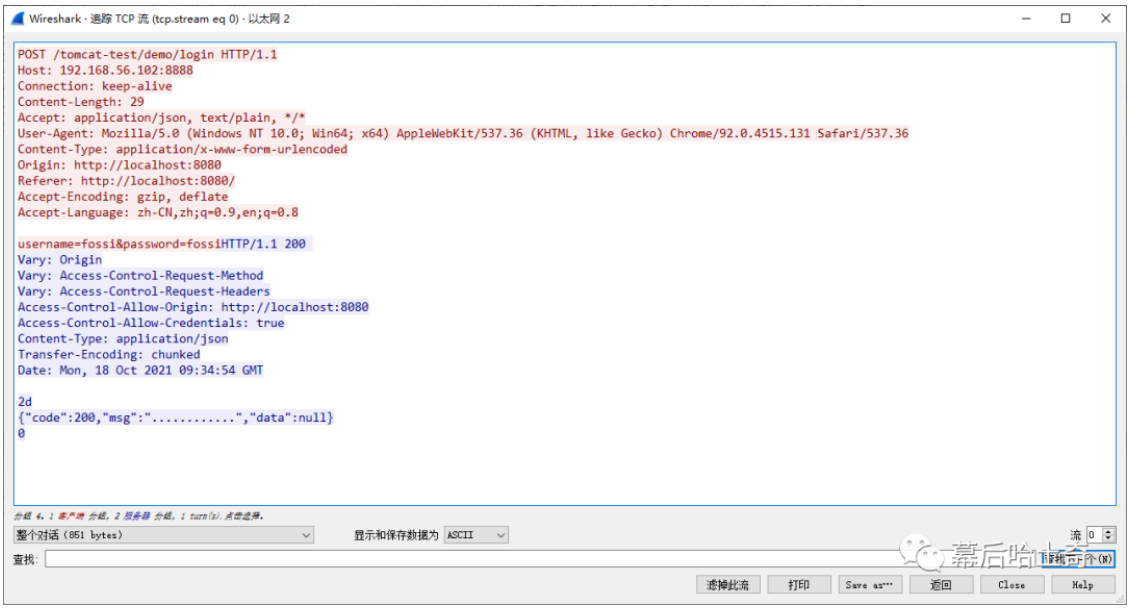
可以看出来，浏览器已经很方便了，它对请求报文和响应报文进行了一个人性化转换，实际的报文是不是如我们上面所说的那种结构呢，我决定用wireshark抓包来看看。

四、抓包验证

好了，我已经顺利完成了抓包，我们先来看看这个过程是如何的。



本次连接中，首先经过TCP三次握手，然后客户端向服务端发起注册请求，随后服务端响应200状态码，接口调用成功，具体我们来看下报文：



首先，红色部分为请求报文，蓝色部分为响应报文。

由于是POST接口，我们填写好用户名密码后，提交的是一个form表单，而不是常用的json，不过这个原理是一样的，请求报文还是分为请求头和请求体两部分，中间用空行分隔。

蓝色部分，分为响应头以及响应体，中间由空行分隔，第一行是协议版本和200状态码。

注册结果的响应体为：

```
{"code":200,"msg":"............","data":null}
```

前端可通过响应体中的code是否为200判断是否注册成功，这里的200是业务级别的结果，代码也可以约定为0是成功，而HTTP响应状态码的200是HTTP定义的标准状态码，我们要注意区分，关于状态码我们暂时还未详细说明到，稍安勿躁。

不过我有个问题，请问这里的2d和0分别是什么意思呢?

```
2d
{"code":200,"msg":"............","data":null}
0
```

后续文章中我们会揭晓。