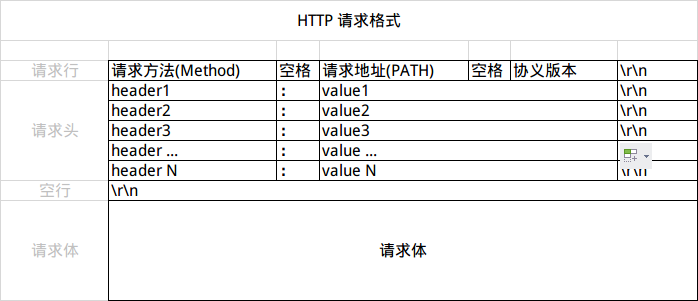
# **你应该知道的HTTP基础知识**

本文主要内容：

* HTTP请求报文格式
* HTTP响应报文格式
* Header
* 请求体的3种形式
* 推荐调试工具
* HTTP的组成图示

## 1. HTTP请求报文格式

HTTP 的请求报文分为三个部分 **请求行**、**请求头**和**请求体**，格式如图：



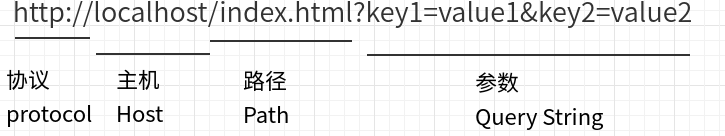
HTTP请求报文格式

注：部分文章也将HTTP请求报文分为两部分请求头和请求体，请求头的第一行为请求行。

#### 1.1 请求行

请求行（Request Line）分为三个部分：**请求方法**、**请求地址**和**协议及版本**，以CRLF(\r\n)结束。  
HTTP/1.1 定义的请求方法有8种：**GET、POST、PUT、DELETE、PATCH、HEAD、OPTIONS、TRACE**,最常的两种GET和POST，如果是RESTful接口的话一般会用到GET、POST、DELETE、PUT。

在了解请求地址之前，先了解一下URL的构成：

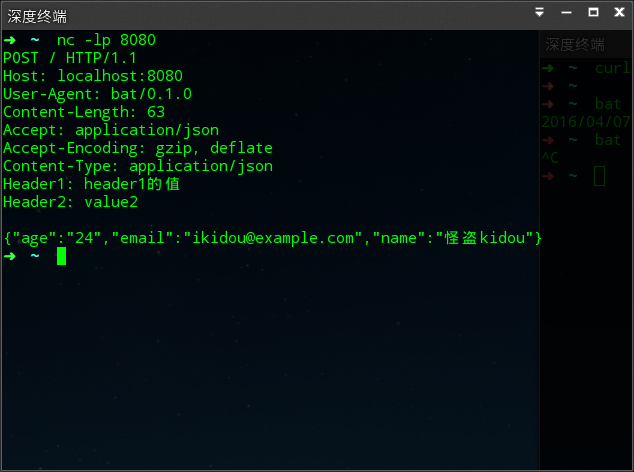


URL.png

PATH是URL主机以后的部分，即包含了Query String，如：

| **URL** | **PATH** |
| --- | --- |
| [http://http://www.jianshu.com/](http://http://www.jianshu.com/" \t "_blank) | / |
| [http://localhost:8080/index.php?id=1234](http://localhost:8080/index.php?id=1234" \t "_blank) | /index.php?id=1234 |
| [http://weibo.com/920507888/home](http://weibo.com/920507888/home" \t "_blank) | /920507888/home |

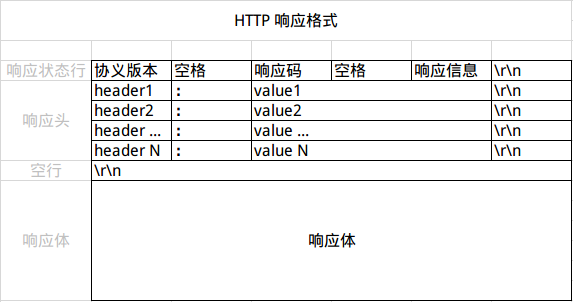
请求报文示例：



HTTP请求报文格式示例

## 2. HTTP响应报文格式

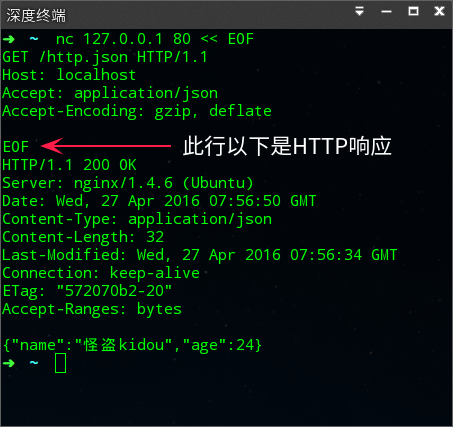
HTTP响应的格式上除状态行(第一行)与请求的请求行不一样以外，其它就格式而言是一样的，但排除状态行和请求行的区别，从Header上还是能区分出HTTP请求和HTTP响应的，怎么区分就要看前面的常见Header啦。



HTTP\_响应

**状态码**（就是上图中的响应码） 的详细可以查看 [HTTP状态码详解](http://tool.oschina.net/commons?type=5" \t "_blank)

为了更直观的展示HTTP响应，下面的例子，自nc 1270.0.1:80 << EOF到EOF之间，是一个简单的HTTP请求.



HTTP请求和响应

## 3. Header

Header可用于传递一些附加信息，格式： 键: 值，注意**冒号后面有一个空格**！如：

Content-Length: 1024

Content-Type: text/plain

#### 3.1 请求和响应常见通用Header

| **名称** | **作用** |
| --- | --- |
| Content-Type | 请求体/响应体的类型，如：text/plain、application/json |
| Accept | 说明接收的类型，可以多个值，用,(半角逗号)分开 |
| Content-Length | 请求体/响应体的长度，单位字节 |
| Content-Encoding | 请求体/响应体的编码格式，如gzip,deflate |
| Accept-Encoding | 告知对方我方接受的Content-Encoding |
| ETag | 给当前资源的标识，和Last-Modified、If-None-Match、If-Modified-Since配合，用于缓存控制 |
| Cache-Control | 取值为一般为no-cache或max-age=XX，XX为个整数，表示该资源缓存有效期(秒) |

#### 3.2 常见请求Header

| **名称** | **作用** |
| --- | --- |
| Authorization | 用于设置身份认证信息 |
| User-Agent | 用户标识，如：OS和浏览器的类型和版本 |
| If-Modified-Since | 值为上一次服务器返回的 Last-Modified 值，用于确认某个资源是否被更改过，没有更改过(304)就从缓存中读取 |
| If-None-Match | 值为上一次服务器返回的 ETag 值，一般会和If-Modified-Since一起出现 |
| Cookie | 已有的Cookie |
| Referer | 表示请求引用自哪个地址，比如你从页面A跳转到页面B时，值为页面A的地址 |
| Host | 请求的主机和端口号 |

#### 3.3 常见响应Header

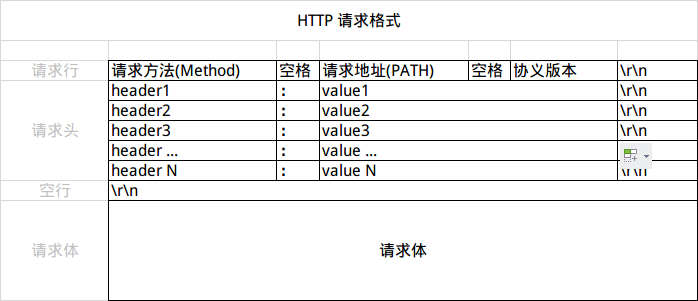
| **名称** | **作用** |
| --- | --- |
| Date | 服务器的日期 |
| Last-Modified | 该资源最后被修改时间 |
| Transfer-Encoding | 取值为一般为chunked，出现在Content-Length不能确定的情况下，表示服务器不知道响应版体的数据大小，一般同时还会出现Content-Encoding响应头 |
| Set-Cookie | 设置Cookie |
| Location | 重定向到另一个URL，如输入浏览器就输入baidu.com回车，会自动跳到 [https://www.baidu.com](https://www.baidu.com" \t "_blank) ，就是通过这个响应头控制的 |
| Server | 后台服务器 |

## 4. 请求体的3种形式

根据应用场景的不同，HTTP请求的请求体有三种不同的形式。

#### 第一种：

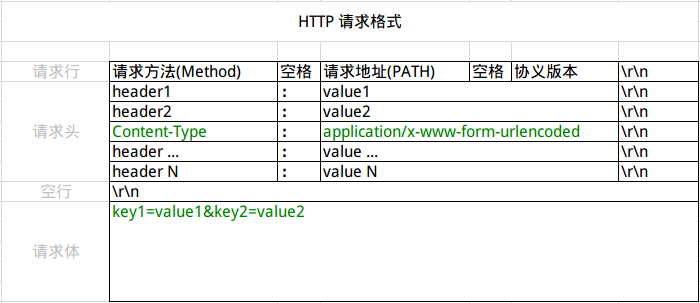
移动开发者常见的，请求体是任意类型，服务器不会解析请求体，请求体的处理需要自己解析，如 POST JSON时候就是这类。



HTTP请求,形式一

#### 第二种：

第二种和第三种都有固定格式的，是服务器端开发人员最先了解到的两种。这里的格式要求就是URL中Query String的格式要求：多个键值对之间用&连接，键与值之前用=连接，且只能用ASCII字符，非ASCII字符需使用UrlEncode编码。



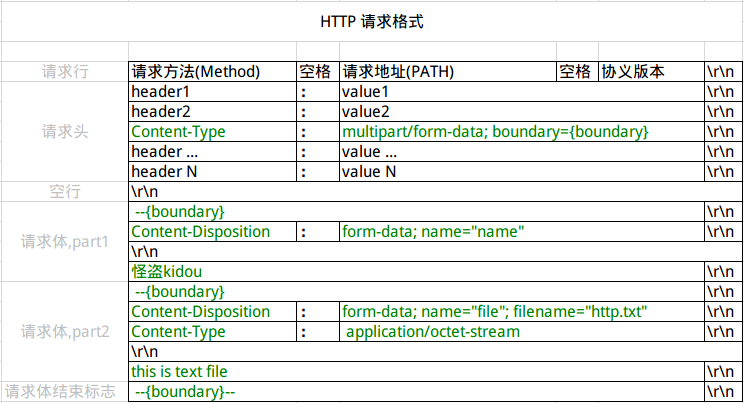
HTTP请求,形式二



第二种示例

#### 第三种：

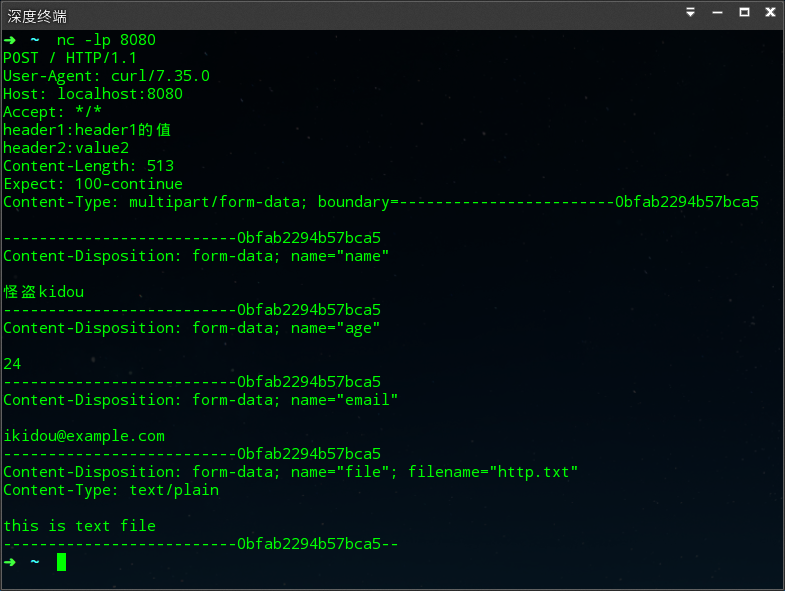
第三种请求体的请求体被分成为多个部分，**文件上传**时会被使用，这种格式最先应该是被用于邮件传输中，每个字段/文件都被boundary（Content-Type中指定）分成单独的段，每段以-- 加 boundary开头，然后是该段的描述头，描述头之后空一行接内容，请求结束的标制为boundary后面加--，结构见下图：



HTTP请求,形式三

区分是否被当成文件的关键是Content-Disposition是否包含filename，因为文件有不同的类型，所以还要使用Content-Type指示文件的类型，如果不知道是什么类型取值可以为application/octet-stream表示该文件是个二进制文件，如果不是文件则Content-Type可以省略。

下图为一个带有文件的上传的请求体原文：



第三种请求体示例

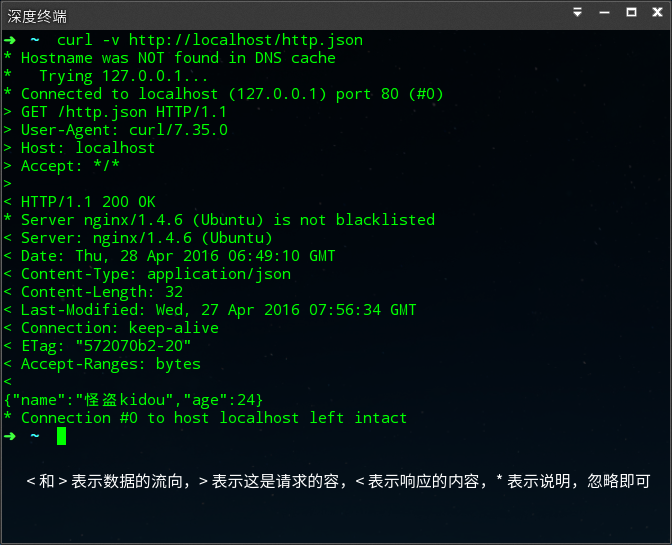
**注意：**  
第二、三种请求体需求配合特定的Content-Type请求头，如：  
第二种配合Content-Type：application/x-www-form-urlencoded  
第三种配合Content-Type： multipart/form-data; boundary={boundary} ，  
\*上面的form-data也可以是mixed、alternative、digest、parallel，但我只用到了form-data  
如果两者没有相配合，那么服务器不会解析请求体，也就是说只会当成第一种情况！

**表单或者模拟表单** 指的就是第二种和第三种(multipart/form-data)

## 5. 推荐调试工具

#### 5.1 cURL

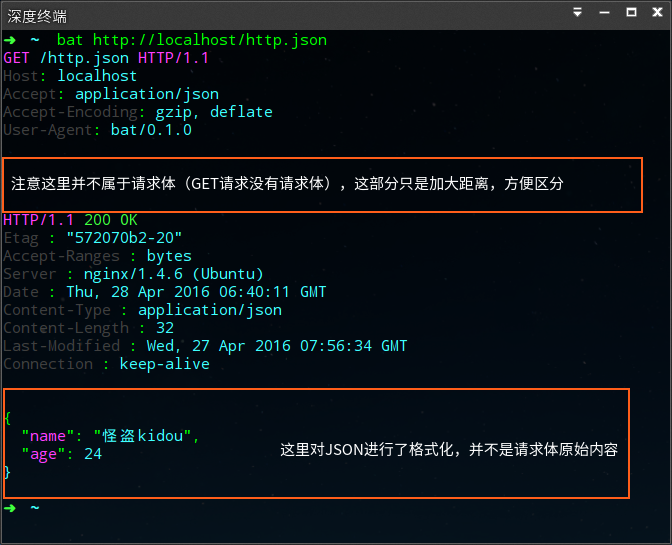
[cURL](https://curl.haxx.se/" \t "_blank) 相当强大命令行工具，基本上你知道的上层协议它都支持，具体使用方法就自行发现了，下图为发起一个HTTP GET请求并打印请求和响应的详细内容。



cURL截图

#### 5.2 bat

[bat](https://github.com/astaxie/bat" \t "_blank) 是[astaxie](https://github.com/astaxie" \t "_blank)(谢孟军) 用Golang开发的类似cURL的命令行API调试工具,可以方便的打印出HTTP请求和响应，还能高亮Header、格式化JSON等功能，非常好用，API调式神器。



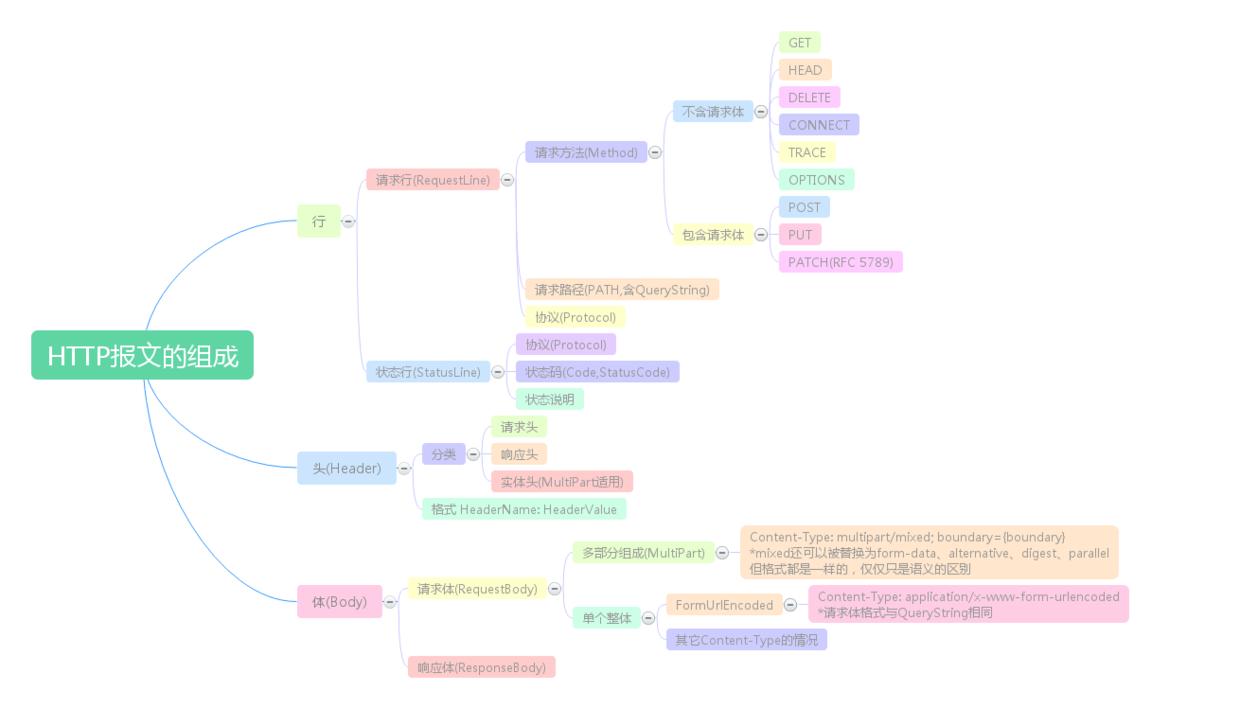
bat截图

#### 5.3 nc

nc 是 netcat 的简写，被称为“网络工具中的瑞士军刀”，不过我个人是把它当成Socket用，经常使用它来打印各种请求，当然它的作用可不只是这样，你也可以用它发起各式各样的请求，以前调式POP3也是用的它，只不过请求报文得自己写，上面有个图“HTTP请求和响应”就是用nc完成的。

## 6. HTTP的组成图示

感觉写得有点乱，如果按下图的结构写会不会更好呢?



HTTP报文组成.png