在开发中，我们使用的比较多的HTTP请求方式基本上就是GET、POST。其中GET用于从服务器获取数据，POST主要用于向服务器提交一些表单数据，例如文件上传等。而我们在使用HTTP请求时中遇到的比较麻烦的事情就是构造文件上传的HTTP报文格式，这个格式虽说也比较简单，但也比较容易出错。今天我们就一起来学习HTTP POST的报文格式以及通过[Java](http://lib.csdn.net/base/17" \t "_blank" \o "Java EE知识库)来模拟文件上传的请求

首先我们来看一个POST的报文请求，然后我们再来详细的分析它。

**POST报文格式**

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. POST /api/feed/ HTTP/1.1
2. Accept-Encoding: gzip
3. Content-Length: 225873
4. Content-Type: multipart/form-data; boundary=OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp
5. Host: www.myhost.com
6. Connection: Keep-Alive
8. --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp
9. Content-Disposition: form-data; name="lng"
10. Content-Type: text/plain; charset=UTF-8
11. Content-Transfer-Encoding: 8bit
13. 116.361545
14. --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp
15. Content-Disposition: form-data; name="lat"
16. Content-Type: text/plain; charset=UTF-8
17. Content-Transfer-Encoding: 8bit
19. 39.979006
20. --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp
21. Content-Disposition: form-data; name="images"; filename="/storage/emulated/0/Camera/jdimage/1xh0e3yyfmpr2e35tdowbavrx.jpg"
22. Content-Type: application/octet-stream
23. Content-Transfer-Encoding: binary
25. 这里是图片的二进制数据
26. --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp--

这里我们提交的是经度、纬度和一张图片（图片数据比较长，而且比较杂乱，这里省略掉了）。

**格式分析**

**请求头分析**

我们先看报文格式中的第一行：

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. POST /api/feed/ HTTP/1.1

这一行就说明了这个请求的请求方式，即为POST方式，要请求的子路径为/api/feed/,例如我们的服务器地址为www.myhost.com,然后我们的这个请求的完整路径就是www.myhost.com/api/feed/，最后说明了HTTP协议的版本号为1.1。

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. Accept-Encoding: gzip
2. Content-Length: 225873
3. Content-Type: multipart/form-data; boundary=OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp
4. Host: www.myhost.com
5. Connection: Keep-Alive

这几个header的意思分别为服务器返回的数据需要使用gzip压缩、请求的内容长度为225873、内容的类型为"multipart/form-data"、请求参数分隔符(boundary)为OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp、请求的根域名为www.myhost.com、HTTP连接方式为持久连接( Keep-Alive)。

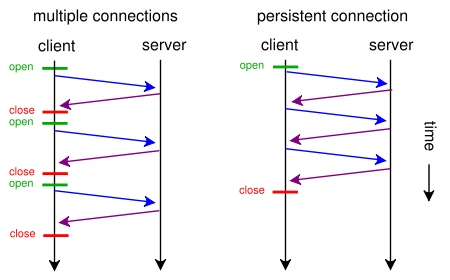
其中这里需要注意的一点是分隔符，即boundary。boundary用于作为请求参数之间的界限标识，例如参数1和参数2之间需要有一个明确的界限，这样服务器才能正确的解析到参数1和参数2。但是分隔符并不仅仅是boundary，而是下面这样的格式：-- + boundary。例如这里的boundary为OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp，那么参数分隔符则为:

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp

不管boundary本身有没有这个"--"，这个"--"都是不能省略的。

我们知道HTTP协议采用“请求-应答”模式，当使用普通模式，即非KeepAlive模式时，每个请求/应答客户和服务器都要新建一个连接，完成之后立即断开连接（HTTP协议为无连接的协议）；当使用Keep-Alive模式（又称持久连接、连接重用）时，Keep-Alive功能使客户端到服务器端的连接持续有效，当出现对服务器的后续请求时，Keep-Alive功能避免了建立或者重新建立连接。



如上图中，左边的是关闭Keep-Alive的情况，每次请求都需要建立连接，然后关闭连接；右边的则是Keep-Alive，在第一次建立请求之后保持连接，然后后续的就不需要每次都建立、关闭连接了，启用Keep-Alive模式肯定更高效，性能更高，因为避免了建立/释放连接的开销。

http 1.0中默认是关闭的，需要在http头加入"Connection: Keep-Alive"，才能启用Keep-Alive；http 1.1中默认启用Keep-Alive，如果加入"Connection: close "，才关闭。目前大部分浏览器都是用http1.1协议，也就是说默认都会发起Keep-Alive的连接请求了，所以是否能完成一个完整的Keep- Alive连接就看服务器设置情况。

**请求实体分析**

请求实体其实就是HTTP POST请求的参数列表，每个参数以请求分隔符开始，即-- + boundary。例如下面这个参数。

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp
2. Content-Disposition: form-data; name="lng"
3. Content-Type: text/plain; charset=UTF-8
4. Content-Transfer-Encoding: 8bit
6. 116.361545

上面第一行为--OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp，也就是--加上boundary内容，最后加上一个换行 （这个换行不能省略），换行的字符串表示为"\r\n"。第二行为Content-Disposition和参数名,这里的参数名为lng，即经度。Content-Disposition就是当用户想把请求所得的内容存为一个文件的时候提供一个默认的文件名，这里我们不过多关注。第三行为Content-Type，即WEB 服务器告诉浏览器自己响应的对象的类型，还有指定字符编码为UTF-8。第四行是描述的是消息请求(request)和响应(response)所附带的实体对象(entity)的传输形式，简单文本数据我们设置为8bit，文件参数我们设置为binary就行。然后添加两个换行之后才是参数的具体内容。例如这里的参数内容为116.361545。

注意这里的每行之间都是使用“\r\n”来换行的，最后一行和参数内容之间是两个换行。文件参数也是一样的格式，只是文件参数的内容是字节流。

这里要注意一下，普通文本参数和文件参数有如下两个地方的不同，因为其内容本身的格式是不一样的。

普通参数：

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. Content-Type: text/plain; charset=UTF-8
2. Content-Transfer-Encoding: 8bit

文件参数：

[plain] [view plain](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/bboyfeiyu/article/details/41863951" \t "_blank" \o "copy)

1. Content-Type: application/octet-stream
2. Content-Transfer-Encoding: binary

参数实体的最后一行是: --加上boundary加上--，最后换行，这里的 格式即为: --OCqxMF6-JxtxoMDHmoG5W5eY9MGRsTBp--。