# OSI七层与TCP/IP五层网络架构

<https://www.2cto.com/net/201310/252965.html>

<http://blog.csdn.net/gavin_john/article/details/53186570>

# TCP、UDP和Socket

# 常见HTTP错误代码

<http://blog.csdn.net/xinxin19881112/article/details/6565823>

404 请求的页面不存在

200 服务器成功返回网页

503 服务不可用

1XX 临时响应

2XX 成功处理了请求的状态代码

## 304

<https://www.douban.com/note/161120791/>

304 的标准解释是：Not Modified 客户端有缓冲的文档并发出了一个条件性的请求（一般是提供If-Modified-Since头表示客户只想比指定日期更新的文档）。服务器告诉客户，原来缓冲的文档还可以继续使用。

如果客户端发起一个文件请求的时候，发现缓存的文件有Last modified，那么在请求中会包含if modified since，这个时间就是缓存文件的最后修改时间。因此，如果请求中包含 If Modified Since，就说明已经有缓存在客户端。只要判断这个时间和当前请求的文件的修改时间就可以确定是返回 304 还是 200 。

对于静态文件，如图片，css等，服务器会自动完成Last modified和if modified since的比较，完成缓存或者更新。

# HTTP请求报文

## HTTP请求行、请求头、请求体详解

<http://blog.csdn.net/u010256388/article/details/68491509>

http请求由三部分组成：请求行+请求头+请求体





①是请求方法，GET和POST是最常见的HTTP方法，除此以外还包括DELETE、HEAD、OPTIONS、PUT、TRACE。不过，当前的大多数浏览器只支持GET和POST，Spring 3.0提供了一个HiddenHttpMethodFilter，允许你通过“\_method”的表单参数指定这些特殊的HTTP方法（实际上还是通过POST提交表单）。服务端配置了HiddenHttpMethodFilter后，Spring会根据\_method参数指定的值模拟出相应的HTTP方法，这样，就可以使用这些HTTP方法对处理方法进行映射了。

②为请求对应的URL地址，它和报文头的Host属性组成完整的请求URL

③是协议名称及版本号。

④是HTTP的报文头，报文头包含若干个属性，格式为“属性名:属性值”，服务端据此获取客户端的信息。

⑤是报文体，它将一个页面表单中的组件值通过param1=value1&param2=value2的键值对形式编码成一个格式化串，它承载多个请求参数的数据。不但报文体可以传递请求参数，请求URL也可以通过类似于“/chapter15/user.html? param1=value1&param2=value2”的方式传递请求参数。

## 常见HTTP请求报文头属性

1. Accept：请求报文可通过一个“Accept”报文头属性告诉服务端 客户端接受什么类型的响应。

Accept属性的值可为一个或多个MIME类型（MIME：<http://en.wikipedia.org/wiki/MIME_type>  ）

1. Cookie：客户端的Cookie就是通过这个报文头属性传给服务端的，如：

Cookie: $Version=1; Skin=new;jsessionid=5F4771183629C9834F8382E23BE13C4C

服务端怎么知道客户端的请求属于哪一个session，就是通过jsessionid判断的。

1. Referer：表示这个请求是从哪个URL过来的
2. Cache-Control：对缓存进行控制，如一个请求希望响应返回的内容在客户端要被缓存一年，或不希望被缓存就可以通过这个报文头达到目的。如：

Cache-Control：no-cache （服务端将对请求返回的响应内容不再客户端进行缓存）

1. Accept-Language：客户端本地化信息
2. Content-Length：请求报文体长度
3. Content-Type：客户端发送的数据格式
4. User-Agent：产生请求的浏览器类型

HttpServletRequest.getSession()的内部机制就是获得请求报文头中Cookie属性的JSESSIONID值。

# HTTP响应报文

由三部分组成：响应行+响应头+响应体



①报文协议及版本；

②状态码及状态描述；

③响应报文头，也是由多个属性组成；

④响应报文体，即我们真正要的“干货”。

和请求报文相比，响应报文多了一个“响应状态码”，它用于告诉客户端此次请求处理结果。

HTTP响应状态码包含5中状态：

1. 1xx 消息，一般是告诉客户端，请求已经收到了，正在处理，别急...
2. 2xx 处理成功，一般表示：请求收悉、我明白你要的、请求已受理、已经处理完成等信息.
3. 3xx 重定向到其它地方。它让客户端再发起一个请求以完成整个处理。
4. 4xx 处理发生错误，**责任在客户端**，如客户端的请求一个不存在的资源，客户端未被授权，禁止访问等。
5. 5xx 处理发生错误，**责任在服务端**，如服务端抛出异常，路由出错，HTTP版本不支持等。

状态码参见：<http://en.wikipedia.org/wiki/List_of_HTTP_status_codes>

HttpServletResponse.setStatus(int sc)：可设置响应状态码。

## 常见的HTTP响应报文头属性

1. Cache-Control：响应输出到客户端后，服务端通过该报文头属告诉客户端如何控制响应内容的缓存
2. ETag
3. Location：我们在JSP中让页面Redirect到一个某个A页面中，其实是让客户端再发一个请求到A页面，这个需要Redirect到的A页面的URL，其实就是通过响应报文头的Location属性告知客户端的，如下的报文头属性，将使客户端redirect到iteye的首页中，如：

Location: <http://www.iteye.com>，这个响应报文属性将使客户端再次发送请求到指定url（即重定向）

1. Set-Cookie：服务端可以设置客户端的Cookie，其原理就是通过这个响应报文头属性实现的，如：

Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1

更多其它的HTTP响应头报文，参见：<http://en.wikipedia.org/wiki/List_of_HTTP_header_fields>

HttpServletResponse.setHeader(String name,String value)：设置响应报文头属性。

addCookie，sendRedirect，方法内部都是修改了响应报文头。

# 四种常见的 POST 提交数据方式

<https://imququ.com/post/four-ways-to-post-data-in-http.html>

HTTP/1.1 协议规定的 HTTP 请求方法有 OPTIONS、GET、HEAD、POST、PUT、DELETE、TRACE、CONNECT 这几种。其中 POST 一般用来向服务端提交数据.

## application/x-www-form-urlencoded

最常见的 POST 提交数据的方式了。浏览器的原生 <form> 表单，如果不设置 enctype 属性，那么最终就会以 application/x-www-form-urlencoded 方式提交数据。

请求类似于下面这样（无关的请求头在本文中都省略掉了）：

POST http://www.example.com HTTP/1.1

Content-Type: application/x-www-form-urlencoded;charset=utf-8

title=test&sub%5B%5D=1&sub%5B%5D=2&sub%5B%5D=3

提交的数据按照 key1=val1&key2=val2 的方式进行编码，key 和 val 都进行了 URL 转码。

很多时候，我们用 Ajax 提交数据时，也是使用这种方式。

## multipart/form-data

使用表单上传文件时，必须让 <form> 表单的 enctype 等于 multipart/form-data。

POST http://www.example.com HTTP/1.1

Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryrGKCBY7qhFd3TrwA

------WebKitFormBoundaryrGKCBY7qhFd3TrwA

Content-Disposition: form-data; name="text"

title

------WebKitFormBoundaryrGKCBY7qhFd3TrwA

Content-Disposition: form-data; name="file"; filename="chrome.png"

Content-Type: image/png

PNG ... content of chrome.png ...

------WebKitFormBoundaryrGKCBY7qhFd3TrwA—

首先生成了一个 boundary 用于**分割**不同的字段，为了避免与正文内容重复，boundary 很长很复杂。然后 Content-Type 里指明了数据是以 **multipart/form-data** 来编码，本次请求的 boundary 是什么内容。消息主体里按照字段个数又分为多个结构类似的部分，每部分都是以 --boundary 开始，紧接着是**内容描述信息**，然后是回车，最后是字段具体内容（**文本或二进制**）。如果传输的是文件，还要包含文件名和文件类型信息。消息主体最后以 --boundary-- 标示结束。

这种方式一般用来上传文件。

上面提到的这两种 POST 数据的方式，都是浏览器原生支持的，而且现阶段标准中原生 <form> 表单也只支持这两种方式（通过 <form> 元素的 enctype 属性指定，默认为 application/x-www-form-urlencoded。其实 enctype 还支持 text/plain，不过用得非常少）。

## applation/json

JSON 格式支持比键值对复杂得多的结构化数据，这一点也很有用。可以方便的提交复杂的结构化数据，特别适合 RESTful 的接口。

POST http://www.example.com HTTP/1.1 Content-Type: application/json;charset=utf-8

{"title":"test","sub":[1,2,3]}

## test/xml

XML-RPC（XML Remote Procedure Call）。它是一种使用 HTTP 作为传输协议，XML 作为编码方式的远程调用规范。