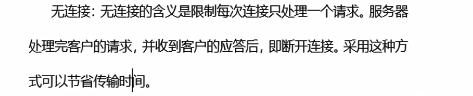
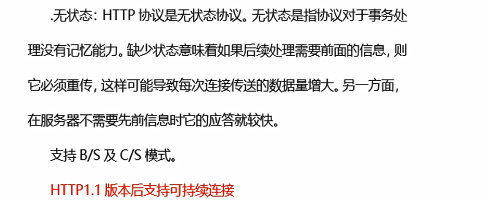
# HTTP协议

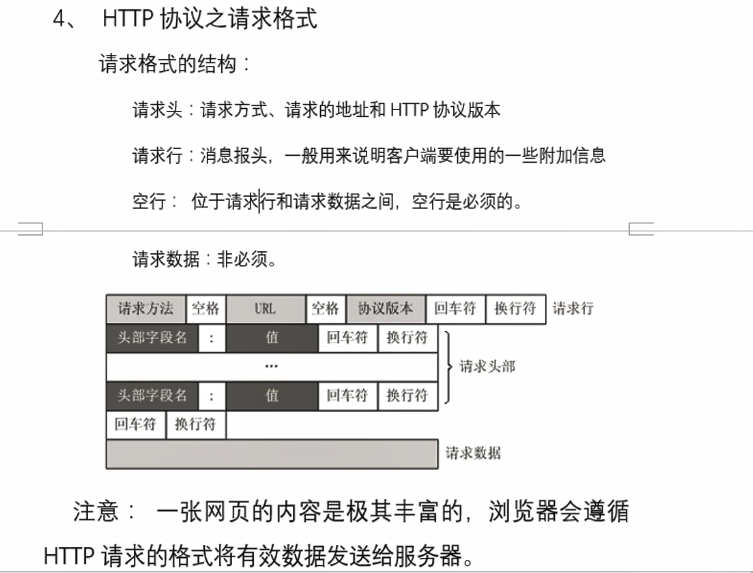
HTTP超文本传输协议，规范了服务器和浏览器双方传输的数据格式，没有协议也能传输但是会阻碍互联网的发展，HTTP的特点是简单、快速、灵活，无状态（对事物处理没有记忆能力，为了解决无状态的问题，引入了Cookie技术。），无连接（每次连接只处理一次请求，在HTTP1.1版本之后支持可持续连接），可以传输任意类型数据（由contentType属性来标记）

C:\Users\孙哈哈\AppData\Local\Temp\WeChat Files\d4f4e83b7a93e053607599333b8236b.png

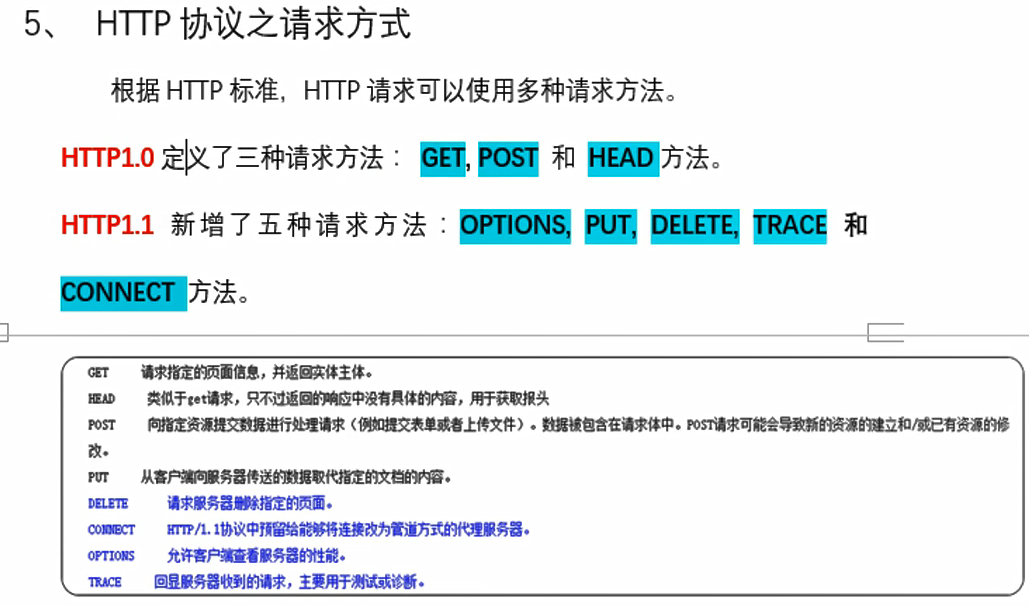




## 请求格式

http协议请求格式内容包括 请求头（请求方法 请求地址 协议版本） 请求体（客户端附加信息支持的格式 cookie等） 空格 请求数据（非必须 get方式为空 post方式可能不为空）





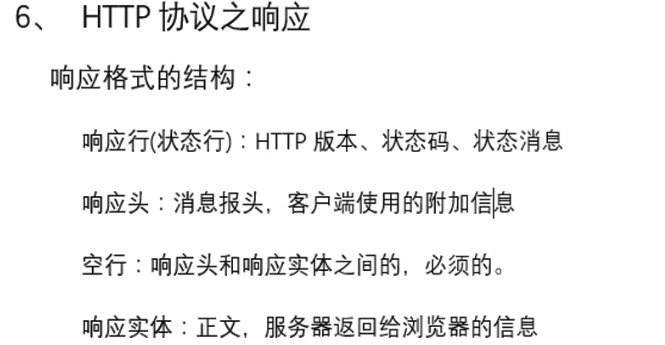
## 响应内容

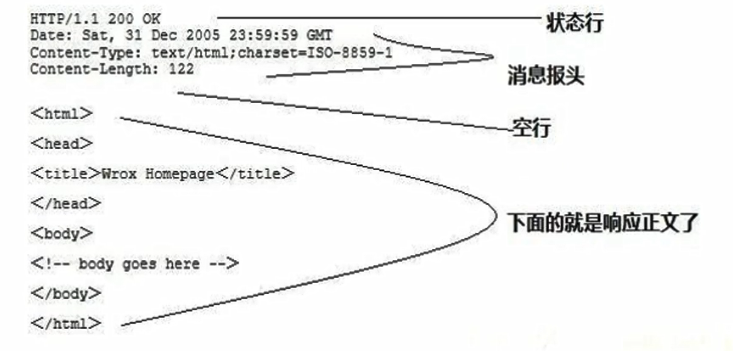
状态行：协议版本 响应状态码 响应状态信息

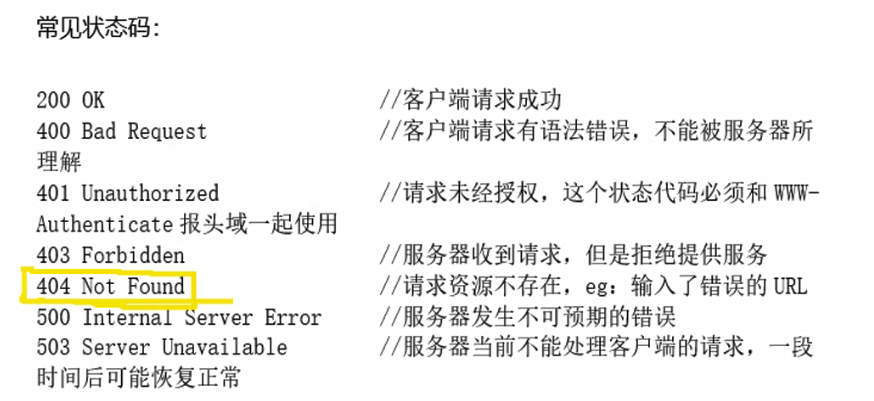
响应头：消息报头，附加说明信息

空行

响应实体

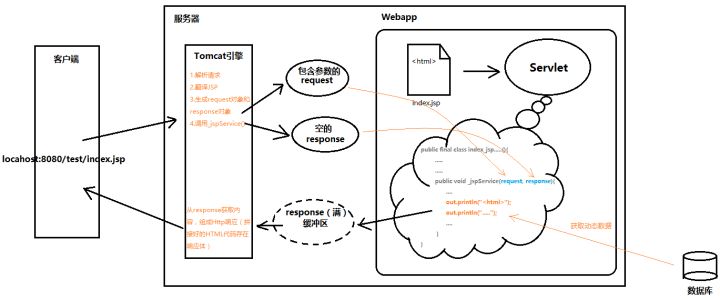






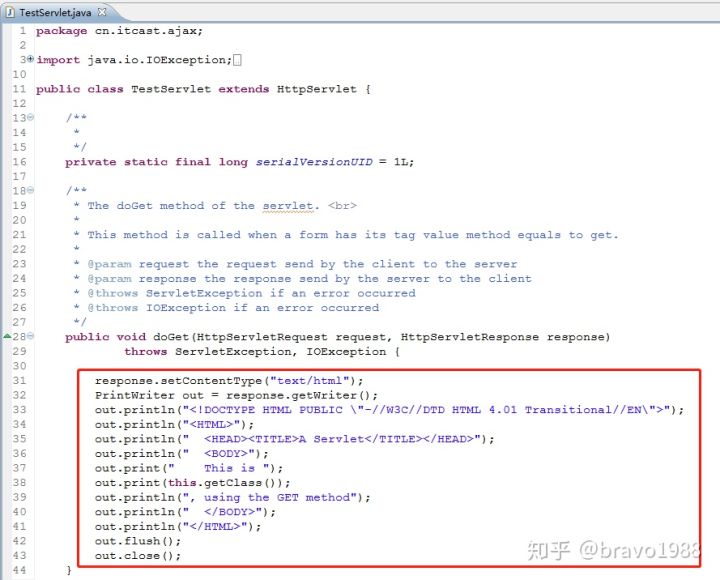
## Servlet与Http

Tomcat将网络请求接收后将请求头、请求体、请求内容封装成request对象和空的response对象传入到对应Servlet中，在执行完处理逻辑后,tomcat会解析response中的内容组装为http响应发送给客户端。



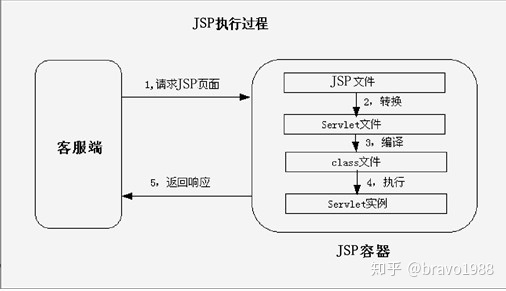
## JSP

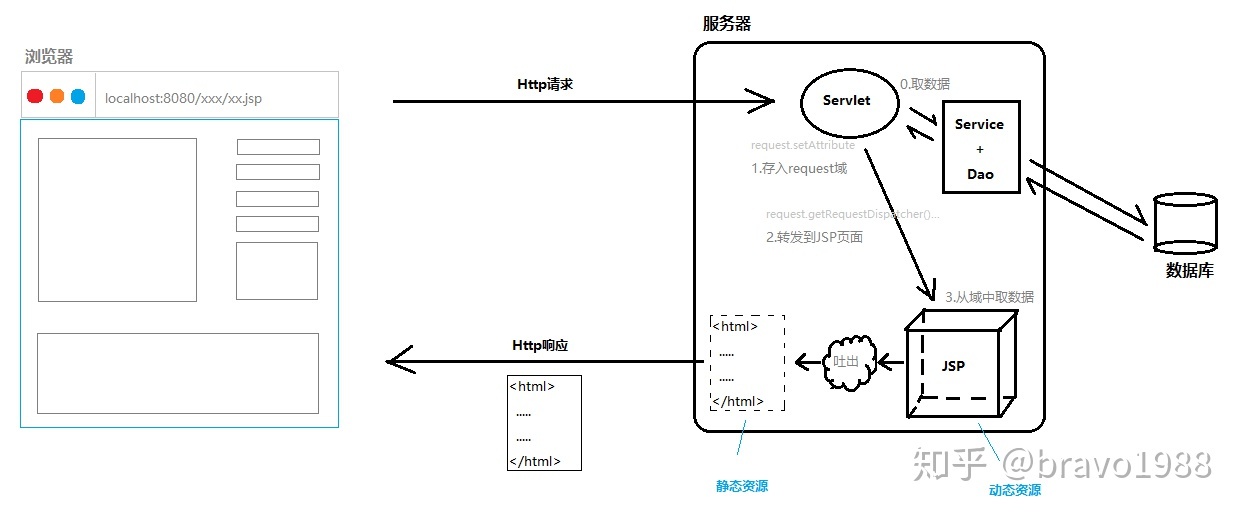
对于动态页面，需要服务器将动态数据嵌入到html页面中，早起servlet用response.write()来逐行输出拼接的代码



这样的方式编写一个页面往往需要成百上千个输出语句非常低效，于是转换思路，将动态内容嵌入到页面中才是正解，于是诞生了jsp,jsp本质还是一个servlet.

每个JSP 页面在第一次被访问时，JSP引擎将它翻译成一个Servlet源程序，接着再把这个Servlet源程序编译成Servlet的class类文件，然后再由WEB容器像调用普通Servlet程序一样的方式来装载和解释执行这个由JSP页面翻译成的Servlet程序。

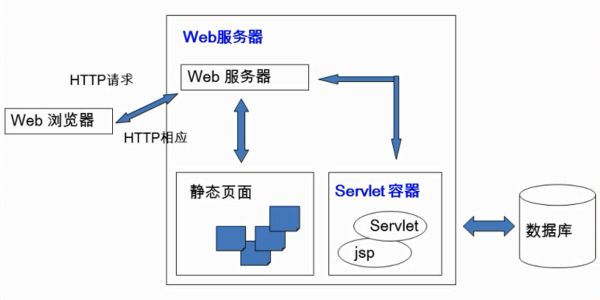




## Tomcat

Tomcat服务器 = Web服务器 + Servlet/JSP容器（Web容器）

Web服务器的作用是接收客户端的请求，给客户端作出响应。但是很明显，服务器不止静态资源呀，所以客户端发起请求后，如果是动态资源，Web服务器不可能直接把它响应回去（比如JSP），因为浏览器只认识静态资源。所以对于JavaWeb程序而言，还需要JSP/Servlet容器，**JSP/Servlet容器的基本功能是把动态资源转换成静态资源。**我们JavaWeb工程师需要使用Web服务器和JSP/Servlet容器，而通常这两者会集于一身，比如Tomcat。



# 网络层传输层协议

## TCP三次握手四次挥手

https://juejin.im/post/5a069b6d51882509e5432656#heading-11

\*ip地址位于网络层

\*mac地址位于数据链路层

\*tcp/ip协议连接需要三次握手（a发送b接受--b确定接受功能正常，b发送a接收--a确定发送和接受都正常，a再次发送b接受--b确定发送正常，双方均通过握手确定了自己的收发功能正常，可以建立连接）

断开需要四次（a发送断开连接请求，b发送确认a进入等待，b再次发送断开连接请求，a发送确认并断开连接）

中断连接端可以是客户端，也可以是服务器端。

第一次挥手：客户端发送一个FIN=M，用来关闭客户端到服务器端的数据传送，客户端进入FIN\_WAIT\_1状态。意思是说"我客户端没有数据要发给你了"，但是如果你服务器端还有数据没有发送完成，则不必急着关闭连接，可以继续发送数据。

第二次挥手：服务器端收到FIN后，先发送ack=M+1，告诉客户端，你的请求我收到了，但是我还没准备好，请继续你等我的消息。这个时候客户端就进入FIN\_WAIT\_2 状态，继续等待服务器端的FIN报文。

第三次挥手：当服务器端确定数据已发送完成，则向客户端发送FIN=N报文，告诉客户端，好了，我这边数据发完了，准备好关闭连接了。服务器端进入LAST\_ACK状态。

第四次挥手：客户端收到FIN=N报文后，就知道可以关闭连接了，但是他还是不相信网络，怕服务器端不知道要关闭，所以发送ack=N+1后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。服务器端收到ACK后，就知道可以断开连接了。客户端等待了2MSL后依然没有收到回复，则证明服务器端已正常关闭，那好，我客户端也可以关闭连接了。最终完成了四次握手。

## TCP与UDP

\*TCP面向有连接请求，是可靠的连接

\*UDP面向无连接请求，是不可靠的连接

# 网络协议基础

//七层协议