Flask是一个Web应用框架，只能处理http请求，如GET、POST等不同类型的HTTP方法

Flask通过装饰器@app.route()来定义路由，指定URL路径和接受的HTTP方法。当客户端发送请求到这些路径时，Flask会调用相应的函数来处理请求。

Flask提供了一个请求对象request，可以通过这个对象获取请求的方法、头部信息、表单数据或JSON数据等信息。这使得开发者能够根据不同的请求类型做出相应的处理。

HTTP协议是建立在TCP协议之上的一种应用

HTTP连接最显著的特点是客户端发送的每次请求都需要服务器回送响应，在请求结束后，会主动释放连接。从建立连接到关闭连接的过程称为“一次连接”。

由于HTTP在每次请求结束后都会主动释放连接，因此HTTP连接是一种“短连接”，要保持客户端程序的在线状态，需要不断地向服务器发起连接请求。通常的做法是即使不需要获得任何数据，客户端也保持每隔一段固定的时间向服务器发送一次“保持连接”的请求，服务器在收到该请求后对客户端进行回复，表明知道客户端“在线”。若服务器长时间无法收到客户端的请求，则认为客户端“下线”，若客户端长时间无法收到服务器的回复，则认为网络已经断开。

Socket是一种TCP连接，Socket连接会通过轮询告诉网络，该连接处于活跃状态。而HTTP连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据。

很多情况下，需要服务器端主动向客户端推送数据，保持客户端与服务器数据的实时与同步。此时若双方建立的是Socket连接，服务器就可以直接将数据传送给客户端；若双方建立的是HTTP连接，则服务器需要等到客户端发送一次请求后才能将数据传回给客户端，因此，客户端定时向服务器端发送连接请求，不仅可以保持在线，同时也是在“询问”服务器是否有新的数据，如果有就将数据传给客户端。

<https://zhuanlan.zhihu.com/p/40777409>网络通信总结（TCP/IP协议、HTTP协议等）

socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口（API），通过Socket，才能使用TCP/IP协议。

Socket其实并不是一个协议，而是为了方便使用TCP/UDP而抽象出来的一层，是位于应用层和传输控制层之间的一组接口。

WebSocket就像HTTP一样，是一个典型的应用层协议。

HTML5 Web Sockets以最小的开销高效地提供了Web连接。相较于经常需要使用推送实时数据到客户端甚至通过维护两个HTTP连接来模拟全双工连接的旧的轮询或长轮询（Comet）来说，这就极大的减少了不必要的网络流量与延迟。

WebSocket是双向通信协议，模拟Socket协议，可以双向发送或接受信息。HTTP是单向的。

WebSocket是需要浏览器和服务器握手进行建立连接的。而http是浏览器发起向服务器的连接，服务器预先并不知道这个连接。

WebSocket在建立握手时，数据是通过HTTP传输的。

WebSocket连接的过程：

客户端发起http请求，经过3次握手后，建立起TCP连接；http请求里存放WebSocket支持的版本号等信息，如：Upgrade、Connection、WebSocket-Version等；

服务器收到客户端的握手请求后，同样采用HTTP协议回馈数据；

客户端收到连接成功的消息后，开始借助于TCP传输信道进行全双工通信。

<https://zhuanlan.zhihu.com/p/542458842>一文梳理HTTP、TCP、Socket和WebSocket的区别和联系

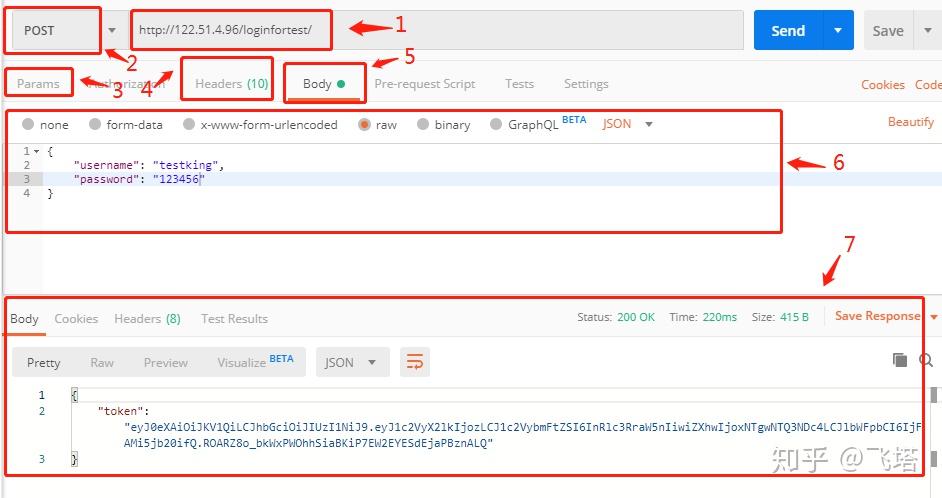
<https://zhuanlan.zhihu.com/p/160910342>WebSocket详解：技术原理、代码演示和应用案例

<https://zhuanlan.zhihu.com/p/51279572>TCP、Socket、WebSocket、HTTP

<https://zhuanlan.zhihu.com/p/113286469>什么是WebSocket，它与HTTP有何不同？

<https://www.cnblogs.com/qiqi715/p/13138589.html>web通讯的四种方式，短轮询、长轮询(comet)、长连接(SSE)、WebSocket

<https://www.cnblogs.com/echoyya/p/14792788.html>客户端与服务器之间双向通讯的5种方式总结(完整代码演示)



请求(Request)包括：

1.请求地址(网络协议+baseUrl+接口地址)

2.请求方式(get/post)

3.请求参数(?key=value)

4.请求头(header,定义接口传输数据的格式,如json或html)

5.请求体(body)

响应数据在接口的返回体中(Response)

状态码：200（成功）、401（未授权）、500（服务端错误）、404（没有该接口）

<https://zhuanlan.zhihu.com/p/113208274>客户端和服务端的通信过程与接口的结构

Http协议是建立在TCP协议基础之上的，当浏览器需要从服务器获取网页数据的时候，会发出一次Http请求。Http会通过TCP建立起一个到服务器的连接通道，当本次请求需要的数据完毕后，Http会立即将TCP连接断开，这个过程是很短的。所以Http连接是一种短连接，是一种无状态的连接。

HTTP协议的主要特点可概括如下：

1.支持客户/服务器模式。

2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。

4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

5.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

URL解析

在WWW上，每一信息资源都有统一的且在网上唯一的地址，该地址就叫URL（Uniform Resource Locator,统一资源定位符），它是WWW的统一资源定位标志，就是指网络地址。HTTP协议工作于客户端-服务端架构之上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

URL由三部分组成：资源类型、存放资源的主机域名、资源文件名。

URL的一般语法格式为(带方括号[]的为可选项)：

protocol :// hostname[:port] / path / [;parameters][?query]#fragment

以下面的URL为例：

http://blog.sina.com.cn/s/blog\_537ad6610102xtb1.html?tj=hist

1、协议部分，代表页面使用的是http协议，在Internet中可以使用多种协议，如HTTP，FTP等等。在"HTTP"后面的“//”为分隔符；

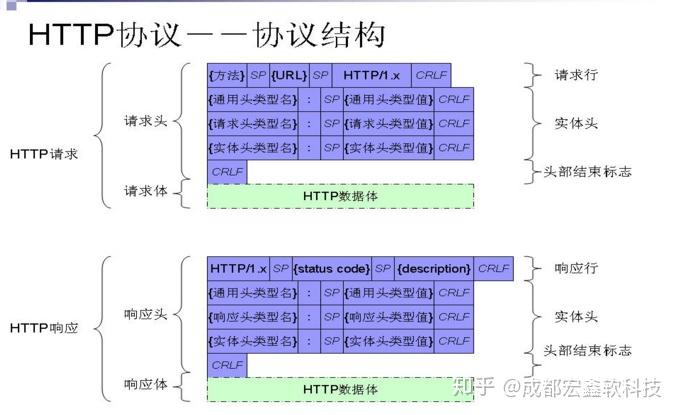
2、域名部分，http://blog.sina.com.cn，也可以使用IP地址作为域名使用如：192.168.55.14:8080，其中8080为端口，域名和端口之间使用“:”作为分隔符。端口不是一个URL必须的部分，如果省略端口部分，将采用默认端口80/tcp；

3、虚拟目录部分，从域名后的第一个“/”开始到最后一个“/”为止，是虚拟目录部分。虚拟目录也不是一个URL必须的部分。本例中的虚拟目录是“/s/”

4、文件名部分：从域名后的最后一个“/”开始到“？”为止，是文件名部分，如果没有“?”,则是从域名后的最后一个“/”开始到“#”为止，是文件部分，如果没有“？”和“#”，那么从域名后的最后一个“/”开始到结束，都是文件名部分。本例中的文件名是“blog\_537ad6610102xtb1.html”。文件名部分也不是一个URL必须的部分，如果省略该部分，则使用默认的文件名

5、锚部分：从“#”开始到最后，都是锚部分。锚部分也不是一个URL必须的部分（可以理解为定位）

6、参数部分：从“？”开始到“#”为止之间的部分为参数部分，又称搜索部分、查询部分。本例中的参数部分为“boardID=5&ID=24618&page=1”。参数可以允许有多个参数，参数与参数之间用“&”作为分隔符。”。参数可以允许有多个参数，参数与参数之间用“&”作为分隔符。



客户端发送一个HTTP请求到服务器的请求消息包括以下格式：

请求行（request line）、请求头部（header）、空行和请求数据四个部分组成。

格式如下：

＜request-line＞(例：GET /index.html HTTP/1.1)

＜headers＞

＜blank line＞

[＜request-body＞](仅post方式有)

post与get的区别

get请求：向服务端获取请求；数据放在URL里，有长度限制；

post请求：向服务端发送请求；数据放在body里，无长度限制；更安全

HTTP响应消息

在接收和解释请求消息后，服务器返回一个HTTP响应消息。

HTTP响应也是由三个部分组成，分别是：状态行、消息报头、响应正文

报文格式：

＜status-line＞(HTTP-Version Status-Code Reason-Phrase CRLF)

＜headers＞

＜blank line＞

[＜response-body＞]

状态码：

1xx：指示信息--表示请求已接收，继续处理

2xx：成功--表示请求已被成功接收、理解、接受

3xx：重定向--要完成请求必须进行更进一步的操作

4xx：客户端错误--请求有语法错误或请求无法实现

5xx：服务器端错误--服务器未能实现合法的请求

常见状态代码、状态描述、说明：

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

<https://zhuanlan.zhihu.com/p/96808201>webservice接口与HTTP接口

<https://zhuanlan.zhihu.com/p/231865779>F12 - 开发者工具详解

<https://zhuanlan.zhihu.com/p/672020173>Chrome 开发者工具保姆级教程

<https://zhuanlan.zhihu.com/p/47697445>Chrome开发者工具详解