什么是Http无状态？Session、Cookie、Token三者之间的区别

一、什么是HTTP无状态？

1.1定义：

HTTP无状态协议，是指协议对于交互性场景没有记忆能力。

1.2举个例子：

在点击一个纯的html网页，请求获取服务器的html文件资源时，每次http请求都会返回同样的信息，因为这个是没有交互的，每一次的请求都是相互独立的。第一个请求和第二个请求也没有先后顺序，返回处理哪个，结果都是同样的资源页面，因为这种场景是无交互的，无论是什么人请求这个地址，服务器都是返回那个相同的响应。

在无交互场景中上面那样，当然也不会有太大的问题。但是对于涉及到动态交互的场景，就显得很尴尬了，何为交互？有来又有往，对于一模一样的两个接口，不同的人在请求第二个接口时可能会基于请求第一个接口的结果而有所不同。

1.3具体场景：

现在我们来想一个复杂的场景，如在购物网站上买一个书包，流程如下：

1. 输入账号密码登陆 /login 用户信息
2. 选择一款你喜欢的书包加入到购物车中 /cart 用户信息，产品信息
3. 购买支付 /pay 用户信息，商品信息，金额信息

所谓的登录只是验证你是否是一个合法用户，若是合法则跳转到信息的页面，不合法则告知用户名密码错误。

但是我们在第一步给服务器发完/login接口后，服务器就忘记了。。。忘记了你这个人，到底有没有经过认证。

所以在添加商品时/cart 你还是需要将你的账号密码和商品信息一起提交给 addCart接口，再让服务器做验证。

第三步同理。

1.4存在的问题：

所以我们说涉及到交互时，情形就完全不一样了，因为这三步是有依赖关系的，第一步验证登录者是一个合法用户，验证通过给你返回200/OK，但是只要服务器给返回了响应，那么一个http的请求和响应就结束了。服务器怎么知道10秒钟之前你刚刚登录过呢？不好意思，服务器不知道你有没有登陆过，他只是对外提供一个登录接口，要想证明你是合法用户必须调/login接口。

第二步，将商品加入到购物车中时，你会调用/cart接口，但是注意，这个行为是和第一步是有关联关系的，是谁将什么物品加入到购物车中了？这个谁，有没有在网站上注册账号呢，是不是一个合法用户呢？所以说在添加购物车的时候，我们还需要将账号密码再次加入到请求参数中，每做一次操作购物车操作时，都需要再把之前已经传输过的账号密码，再反反复复的传输一遍又一遍，这是因为服务器不知道你是不是在20秒之前刚登陆过。

1.5总结：

上面的无状态是指的，无登录状态，即服务器不知道某个用户是否已登录过了。因为愚蠢的服务器不知道客户端是否已登录过了,所以每次都要在交互场景(会话)中请求中带上上一次的请求信息，如账号、密码。明明只需要在/login接口中，才需要对比数据库中的账号密码和客户端传的是否一致来确定合法性。这下在添加购物车中也需要再一次的进行同样的重复且没有必要的操作，即降低了响应速度，又对用户不友好（因为每次都需要填账号，密码）。

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。另外人们常说的"会话"概念则是上面的交互行为的另一种表述方式。

二、让服务器有记忆能力之Cookie、Session

背景：

通过上面我们知道了Http中无状态是一个什么概念，以及在无状态情况下，要进行添加购物车功能，所带来的困难。

前瞻：

客户端与服务器进行动态交互的Web应用程序出现之后，HTTP无状态的特性严重阻碍了这些应用程序的实现，毕竟交互是需要承前启后的，简单的购物车程序也要知道用户到底在之前选择了什么商品。于是，两种用于保持HTTP连接状态的技术就应运而生了，一个是Cookie，而另一个则是Session。HTTP本身是一个无状态的连接协议，为了支持客户端与服务器之间的交互，我们就需要通过不同的技术为交互存储状态，而这些不同的技术就是Cookie和Session了。

Cookie

由于HTTP是一种无状态的协议，服务器***单纯从网络连接上***无从知道客户身份。怎么办呢？就给客户端们颁发一个通行证吧，每人一个，无论谁访问都必须携带自己通行证。这样服务器就能从通行证上确认客户身份了。这就是Cookie的工作原理。

Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用response向客户端浏览器颁发一个Cookie。客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，***浏览器把请求的网址连同该Cookie一同提交给服务器***。服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容。

Cookie的不可跨域名性

很多网站都会使用Cookie。例如，Google会向客户端颁发Cookie，Baidu也会向客户端颁发Cookie。那浏览器访问Google会不会也携带上Baidu颁发的Cookie呢？或者Google能不能修改Baidu颁发的Cookie呢？

答案是否定的。Cookie具有不可跨域名性。根据Cookie规范，浏览器访问Google只会携带Google的Cookie，而不会携带Baidu的Cookie。Google也只能操作Google的Cookie，而不能操作Baidu的Cookie。

Cookie在客户端是由浏览器来管理的。浏览器能够保证Google只会操作Google的Cookie而不会操作Baidu的Cookie，从而保证用户的隐私安全。浏览器判断一个网站是否能操作另一个网站Cookie的依据是域名。Google与Baidu的域名不一样，因此Google不能操作Baidu的Cookie。

Cookie的一个实例

1.在登录网站的时候选择记住密码

2.点击之后观察服务器的相应内容

3.查看Chrome中的Cookie设置

4.观察服务器返回的Cookie内容

5.再次访问时，发现不需要输入密码即可直接登录，观察请求头信息

Cookie记录登录状态的三个方案

cookie并不是单纯为了实现 session机制而生的。而是1993 年，网景公司雇员 Lou Montulli 为了让用户在访问某网站时，进一步提高访问速度，同时也为了进一步实现个人化网络，发明了今天广泛使用的 Cookie。cookie还用一个很广泛的用途就是记住用户的登录账号和密码，这样当用户以后再次需要登录同一个网站或系统的时候就不需要再次填写这两个字段而是直接点击“登录”按钮就好。这就相当于给了一些“甜头”给用户，这就回应了“cookie”这个词语的字面意思了。

> 如果用户是在自己家的电脑上网，登录时就可以记住他的登录信息，下次访问时不需要再次登录，直接访问即可。

实现方法是把登录信息如账号、密码等保存在Cookie中，并控制Cookie的有效期，下次访问时再验证Cookie中的登录信息即可。

保存登录信息有多种方案

- 方案一：最直接的是把用户名与密码都保持到Cookie中，下次访问时检查Cookie中的用户名与密码，与数据库比较。这是一种比较危险的选择，一般不把密码等重要信息保存到Cookie中。

- **方案二：是把密码加密后保存到Cookie中，下次访问时解密并与数据库比较。这种方案略微安全一些。如果不希望保存密码，还可以把登录的时间戳保存到Cookie与数据库中，到时只验证用户名与登录时间戳就可以了。**
- **方案三：只在登录时查询一次数据库，以后访问验证登录信息时不再查询数据库。实现方式是把账号按照一定的规则加密后，连同账号一块保存到Cookie中。下次访问时只需要判断账号的加密规则是否正确即可。**

> 一和二两种方案验证账号时都要查询数据库。
>
> 方案三把账号保存到名为account的Cookie中，把账号连同密钥用MD5算法加密后保存到名为ssid的Cookie中。验证时验证Cookie中的账号与密钥加密后是否与Cookie中的ssid相等。
>
> 提示：该加密机制中最重要的部分为算法与密钥。由于MD5算法的不可逆性，即使用户知道了账号与加密后的字符串，也不可能解密得到密钥。因此，只要保管好密钥与算法，该机制就是安全的。

Session

> 由于网页是一种无状态的连接程序，因此无法得知用户的浏览状态。在网上购物的时，把很多商品加入了购物车，而在结账时网站却不知道你购物车有哪些物品。为了解决这个问题，服务器端就为特定用户创建了特定的session，用于标示并跟踪这个用户，这样才知道购物车里有哪些商品。

- **Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。**
- **客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。**
- **如果说Cookie机制是通过检查客户身上的“通行证”来确定客户身份的话，那么Session机制就是通过检查服务器上的“客户明细表”来确认客户身份。**
- **Session相当于程序在服务器上建立的一份客户档案，客户来访的时候只需要查询客户档案表就可以了。**

Session和Cookie的关系

- cookie **是一个实际存在的、具体的东西，**http **协议中定义在** header **中的字段。**
- session **是一个抽象概念、开发者为了实现中断和继续等操作，将**client**和**server**之间一对一的交互，抽象为“会话”，进而衍生出“会话状态”，也就是** session **的概念。**
- **即**session**描述的是一种通讯会话机制，而**cookie**只是目前实现这种机制的主流方案里面的一个参与者，它一般是用于保存**session ID**。**

Token**的相关讲解见** http://weixin.niurenqushi.com/article/2017-03-20/4794863.html
HMac**介绍：**hmac

参考文章：

https://www.jianshu.com/p/bdc97c096fbc
https://www.cnblogs.com/bellkosmos/p/5237146.html
http://weixin.niurenqushi.com/article/2017-03-20/4794863.html
https://www.cnblogs.com/yupeng/archive/2012/05/24/2515228.html