会话

**会话**：用户打开一个浏览器，点击了很多超链接，访问多个web资源，关闭浏览器，这个过程可以称之为会话；

**有状态会话**：一个同学来过教室，下次再来教室，我们会知道这个同学，曾经来过，称之为有状态会话；

**你能怎么证明你是西开的学生？**

你 西开

1. 发票 西开给你发票
2. 学校登记 西开标记你来过了

**一个网站，怎么证明你来过？**

客户端 服务端

1. 服务端给客户端一个 信件，客户端下次访问服务端带上信件就可以了； cookie //cookie存储在客户端
2. 服务器登记你来过了，下次你来的时候我来匹配你； session //session存储在服务器端

**cookie**

* 客户端技术 （响应，请求）

**session**

* 服务器技术，利用这个技术，可以保存用户的会话信息？ 我们可以把信息或者数据放在Session中！

1. 由于HTTP协议是无状态的协议，所以服务端需要记录用户的状态时，就需要用某种机制来识具体的用户，这个机制就是Session.典型的场景比如购物车，当你点击下单按钮时，由于HTTP协议无状态，所以并不知道是哪个用户操作的，所以服务端要为特定的用户创建了特定的Session，用用于标识这个用户，并且跟踪用户，这样才知道购物车里面有几本书。这个Session是保存在服务端的，有一个唯一标识。在服务端保存Session的方法很多，内存、数据库、文件都有。集群的时候也要考虑Session的转移，在大型的网站，一般会有专门的Session服务器集群，用来保存用户会话，这个时候 Session 信息都是放在内存的，使用一些缓存服务比如Memcached之类的来放 Session。  
2. 思考一下服务端如何识别特定的客户？这个时候Cookie就登场了。每次HTTP请求的时候，客户端都会发送相应的Cookie信息到服务端。实际上大多数的应用都是用 Cookie 来实现Session跟踪的，第一次创建Session的时候，服务端会在HTTP协议中告诉客户端，需要在 Cookie 里面记录一个Session ID，以后每次请求把这个会话ID发送到服务器，我就知道你是谁了。有人问，如果客户端的浏览器禁用了 Cookie 怎么办？一般这种情况下，会使用一种叫做URL重写的技术来进行会话跟踪，即每次HTTP交互，URL后面都会被附加上一个诸如 sid=xxxxx 这样的参数，服务端据此来识别用户。  
3. Cookie其实还可以用在一些方便用户的场景下，设想你某次登陆过一个网站，下次登录的时候不想再次输入账号了，怎么办？这个信息可以写到Cookie里面，访问网站的时候，网站页面的脚本可以读取这个信息，就自动帮你把用户名给填了，能够方便一下用户。这也是Cookie名称的由来，给用户的一点甜头。  
所以，总结一下：  
Session是在服务端保存的一个数据结构，用来跟踪用户的状态，这个数据可以保存在集群、数据库、文件中；  
Cookie是客户端保存用户信息的一种机制，用来记录用户的一些信息，也是实现Session的一种方式。

asDf·

1，session 在服务器端，cookie 在客户端（浏览器）  
2，session 默认被存在在服务器的一个文件里（不是内存）  
3，session 的运行依赖 session id，而 session id 是存在 cookie 中的，也就是说，如果浏览器禁用了 cookie ，同时 session 也会失效（但是可以通过其它方式实现，比如在 url 中传递 session\_id）  
4，session 可以放在 文件、数据库、或内存中都可以。  
5，用户验证这种场合一般会用 session

因此，维持一个会话的核心就是客户端的唯一标识，即 session id

1、Cookie 在客户端（浏览器），Session 在服务器端。

2、Cookie的安全性一般，他人可通过分析存放在本地的Cookie并进行Cookie欺骗。在安全性第一的前提下，选择Session更优。重要交互信息比如权限等就要放在Session中，一般的信息记录放Cookie就好了。

3、单个Cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个Cookie。

4、Session 可以放在 文件、数据库或内存中，比如在使用Node时将Session保存在redis中。由于一定时间内它是保存在服务器上的，当访问增多时，会较大地占用服务器的性能。考虑到减轻服务器性能方面，应当适时使用Cookie。

5、Session 的运行依赖Session ID，而 Session ID 是存在 Cookie 中的，也就是说，如果浏览器禁用了 Cookie，Session 也会失效（但是可以通过其它方式实现，比如在 url 中传递 Session ID）。

6、用户验证这种场合一般会用 Session。因此，维持一个会话的核心就是客户端的唯一标识，即Session ID。

Cookie与Session的具体区别：

1…存储位置不同

通常情况

Cookie的数据信息存放在客户端浏览器上。

Session的数据信息存放在服务器上。

2.存储容量不同

通常情况

单个Cookie保存的数据≤4KB，一个站点最多保存20个Cookie。

对于Session并没有上限，但出于对服务器端的性能考虑，Session内不要存放过多的东西，并设置session删除机制

3.存取方式不同：

Cookie中只能保管String，需要通过编码的方式存取Unicode字符或者二进制数据。运用Cookie难以实现存储略微复杂的信息

4.隐私策略不同

Cookie对客服端是可见的，别有用心的人可以分析放在本地的Cookie上面并进行Cookie欺骗，所以它是不安全的

5.有效期不同

开发可以通过设置Cookie的属性，达到Cookie长期有效的效果

由于Session依赖于名为JSESSIONID的cookie，而Cookie JSESSIONID的过期时间默认为-1，只需关闭窗口该Session就会失效，因而Session不能达到长期有效的效果，就算不依赖Cookie，运用URL地址重写也不能完成，因为假如设置Session的超过时间过长，服务器累计的Session就会越多，越容易导致内存溢出

6.服务器压力不同

Session是保管在服务端的，每个用户都会产生一个Session，假如并发访问的用户十分多，会产生十分多的Session，耗费大量的内存

Cookie是保管在客户端，不占用服务器资源，对于并发用户十分多的网站，cookie是很好的选择

7.浏览器支持不同：

假如客户端浏览器不支持Cookie。

Cookie是需要客户端浏览器支持的。假如客户端禁用了Cookie，或者不支持Cookie，则会话跟踪会失效。关于WAP上的应用，常规的Cookie就派不上用场了。

运用Session需要使用URL地址重写的方式。一切用到Session程序的URL都要进行URL地址重写，否则Session会话跟踪还会失效。关于WAP应用来说，Session+URL地址重写或许是它唯一的选择。

假如客户端支持Cookie。

Cookie 既能够设为本浏览器窗口以及子窗口内有效（把过期时间设为−1），也能够设为一切窗口内有效（把过期时间设为某个大于0的整数）。

Session 只能在本窗口以及其子窗口内有效。假如两个浏览器窗口互不相干，它们将运用两个不同的Session。（IE8下不同窗口Session相干。）

8.跨域支持上的不同

Cookie 支持跨域名访问，例如，将 domain 属性设置为“.biaodianfu.com”，则以“.biaodianfu.com”为后缀的一切域名均能够访问该Cookie。跨域名Cookie如今被普遍用在网络中，例如，Google、Baidu、Sina等。

Session则不会支持跨域名访问。Session仅在它所在的域名内有效。