会话技术是什么

在日常生活中，从拨通电话到挂断电话之间，一连串的你问我答的过程就是一个会话。**而在Web应用中，会话指的是从使用浏览器打开某个网站到关闭浏览器之间，一连串的（浏览器）请求和（服务器）响应的过程**。类似于生活中的打电话过程。

会话过程中存在哪些问题？

在打电话过程中，通话双方会有通话内容，同样，在浏览器与服务器端交互的过程中，也会产生一些数据。例如，在购物网站中，有甲、乙两个用户，甲购买了一个iPhone X，乙购买了一个iPad，服务器需要对他们的购买数据进行分别保存，同样，在结账时，服务器需要判断结账的用户是甲还是乙，以便能够取出对应数据。那么问题来了！服务器如何识别用户的身份？

会话技术，是识别用户身份的技术，包括两种：

> *Cookie技术——将用户的身份数据保存在客户端（浏览器）*
>
> *Session技术——将用户的身份数据保存在服务器*

Cookie是什么？

Cookie使用浏览器进行用户数据的保存这样的一种**客户端**技术！

生活场景：在现实生活中，某些理发店、美容院、商城等实体店，经常会推荐顾客办理或赠送顾客一张会员卡，卡上记录了用户的个人信息（姓名、手机号等）、余额、积分等数据，如果顾客办理了会员卡，以后每次光临这些实体店时，都会携带这张会员卡，实体店也将根据会员卡的类型、消费金额等给顾客相应的优惠。

在Web应用中，Cookie类似于这张会员卡，详细的对应关系如下：

> *实体店——服务器*
>
> *顾客——浏览器*
>
> *会员卡——Cookie（身份标识）*

> *顾客第一次光临实体店，没有会员卡<-------> 浏览器第一次访问服务器，没有cookie*
>
> *实体店办理会员卡，发给顾客<-------> 服务器创建Cookie，发给浏览器*
>
> *顾客把会员卡保存在？<-------> 浏览器把Cookie保存在客户端的磁盘中*

> *顾客以后光临实体店，带上会员卡<-------> 浏览器以后访问服务器，自动带上Cookie*
>
> *实体店根据会员卡给顾客相应的优惠<-------> 服务器根据Cookie（识别用户的身份）给浏览器相应的响应*

1　WEB服务器通过在HTTP响应消息中增加Set-Cookie响应头字段将Cookie信息发送给浏览器，浏览器则通过在HTTP请求消息中增加Cookie请求头字段将Cookie回传给WEB服务器。

2　一个Cookie只能标识一种信息，它至少含有一个标识该信息的名称（NAME）和设置值（VALUE）。

3　Cookie的数据库格式 key=value

4　一个WEB站点可以给一个WEB浏览器发送多个Cookie，一个WEB浏览器也可以存储多个WEB站点提供的Cookie。

5　浏览器一般只允许存放300个Cookie，每个站点最多存放20个Cookie，每个Cookie的大小限制为4K

javax.servlet.http

Class Cookie

java.lang.Object
  └ javax.servlet.http.Cookie

All Implemented Interfaces:
Cloneable

public class **Cookie**
extends Object
implements Cloneable

Implements: Cloneable

创建一个 cookie，cookie 是 servlet 发送到 Web 浏览器的少量信息，这些信息由浏览器保存，然后发送回服务器。cookie 的值可以唯一地标识客户端，因此 cookie 常用于会话管理。

一个 cookie 拥有一个名称、一个值和一些可选属性，比如注释、路径和域限定符、最大生存时间和版本号。一些 Web 浏览器在处理可选属性方面存在 bug，因此有节制地使用这些属性可提高 servlet 的互操作性。

servlet 通过使用HttpServletResponse#addCookie方法将 cookie 发送到浏览器，该方法将字段添加到 HTTP 响应头，以便一次一个地将 cookie 发送到浏览器。浏览器应该支持每台 Web 服务器各20 个 cookie，总共有300 个 cookie，并且可能将每个 cookie 的大小限定为4 KB。

浏览器通过向 HTTP 请求头添加字段将 cookie 返回给 servlet。可使用HttpServletRequest#getCookies方法从请求中获取 cookie。一些 cookie 可能有相同的名称，但却有不同的路径属性。

cookie 影响使用它们的 Web 页面的缓存。HTTP 1.0 不会缓存那些使用通过此类创建的 cookie 的页面。此类不支持 HTTP 1.1 中定义的缓存控件。

创建Cookie

1. **Cookie的API**

a.

Constructor Summary

参数1：cookie的名字
参数2：cookie的值

`Cookie(String name, String value)`
Constructs a cookie with a specified na

void	setComment(String purpose) Specifies a comment that describes a cookie's purpose.
void	setDomain(String pattern) Specifies the domain w
void	setMaxAge(int expiry) Sets the maximum age of the cookie in seconds.
void	setPath(String uri) Specifies a path for the
void	setSecure(boolean flag) Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HT
void	setValue(String newValue) Assigns a new value to a cookie after the cookie is created.
void	setVersion(int v) Sets the version of the cookie protocol this cookie complies with.

设置cookie的最大生存时间，单位是秒

设置cookie的路径，即servlet获取这些cookie时的一些限制

2. 代码示例

a.

```
//创建Cookie
Cookie ck=new Cookie("code", code);
ck.setPath("/");//设置Cookie的路径
ck.setMaxAge(-1);//内存存储，取值有三种：>0有效期，单位秒；=0失效；<0内存存储
response.addCookie(ck);//让浏览器添加Cookie
```

获取Cookie

1. 代码示例

a.

```
//获取所有的Cookie
Cookie[] cks=request.getCookies();
//遍历Cookie
for(Cookie ck:cks){
//检索出自己的Cookie
if(ck.getName().equals("code")){
            //记录Cookie的值
           code=ck.getValue();
           break;
      }
  }
```

如何修改Cookie

只需要保证Cookie的名和路径与修改之前一致即可修改；例如：在服务端将用户浏览器中的某个Cookie修改为失效，只需要创建一个和被修改的Cookie相同名和路径的Cookie，将这个新的Cookie的失效时间为0即可。

```
//创建Cookie
Cookie ck=new Cookie("code", code);
ck.setPath("/");//设置Cookie的路径
ck.setMaxAge(-1);//内存存储，取值有三种：>0有效期，单位秒；=0失效；<0内存存储
response.addCookie(ck);//让浏览器添加Cookie
```

Cookie的生存时间

```
ck.setMaxAge(-1)；设置生成时间
取值说明：
>0有效期，单位秒
=0失效
<0内存存储
```

Cookie的编码与解码(面试题)

中文和英文字符不同，中文属于Unicode字符，在内存中占用4个字符，而英文属于ASCII字符，内存中只占2个字节。Cookie中使用Unicode字符时需要对Unicode字符进行编码，否则会出现乱码,存储会异常！

> *编码使用java.net.URLEncoder类的encode(String str,String encoding)方法*
>
> *解码使用java.net.URLDecoder类的decode(String str,String encoding)方法*

代码如下：
保存：Servlet类

```
1       // 使用中文的 Cookie. name 与 value 都使用 UTF-8 编码.
2       Cookie cookie = new Cookie(
3        URLEncoder.encode("姓名", "UTF-8"),
4        URLEncoder.encode("老刘", "UTF-8"));
5       // 发送到客户端
6       response.addCookie(cookie);
```

读取：jsp页面

```jsp
<%
    if(request.getCookies() != null){
        for(Cookie cc : request.getCookies()){
            String cookieName =
URLDecoder.decode(cc.getName(),"UTF-8");
            String cookieValue =
URLDecoder.decode(cc.getValue(),"UTF-8");
            out.println(cookieName + "=");
            out.println(cookieValue + "; <br/>");
        }
    }
%>
```

Cookie的使用场景

1. 商品的浏览记录
2. 记住用户名（用户再次进入登录页面进行登录时不需要再输入用户名而是自动显示）
3. 免登陆
4. 显示上次登录系统的时间

Cookie案例--记住用户名

1. 功能分析

> *一般的网站只提供记住用户名操作，如果提供自动登录功能，一般给出安全提示，因为cookie数据保存在客户端的磁盘中，不安全，因此像银行账号和密码等之类的敏感数据一般不往cookie中保存.*

总结

> *1.Cookie数据保存在浏览器（磁盘中）*
>
> *2.每个浏览器cookie的大小和个数有限制，而且限制数不太一样*

各浏览器对Cookie有一定的限制，在使用时需要格外注意。

各浏览器之间对cookie的不同限制：

	IE6.0	IE7.0/8.0/9.0+	Opera	FF	Safari	Chrome
cookie个数	每个域为20个	每个域为50个	每个域为30个	每个域为50个	没有个数限制	每个域为53个
cookie大小	4095个字节	4095个字节	4096个字节	4097个字节	4097个字节	4097个字节

> *3.Cookie有作用范围*
>
> 1. *浏览器之间cookie数据不能共享，火狐浏览器中的Cookie不能共享到谷歌中！*
> 2. *在服务端可以设置Cookie起作用作用的域名和路径，通过调用setDomian、setPath方法来设置，最终表现的结果就是：只有用户在浏览器中发起的URL是cookie起作用的路径或子路径，浏览器才会自动在请求头中加入Cookie字段*
>
> ```
> 示例：
> Cookie c = new Cookie(URLEncoder.encode("李旭东"), "cshhy1");
> c.setDomain("localhost");
> //只有用户下次访问路径：
> localhost:8080/shitou1/index的时候，请求头中才会
> //带有cookie(李旭东-cshhy1>)，访问别的路径的时候，则不会。
> c.setPath("/shitou1/index");
> response.addCookie(c);
> ```
>
> *4.由于cookie数据保存在本地磁盘中，因此存在安全隐患，因此应该尽量避免使用Cookie 保存重要的敏感数据，如果非要保存，需要对保存的数据进行某些可逆加密算法进行加密，只要这个加密和解密的工具不泄露，就意味着数据是安全的！对于密码之类的数据，使用不可逆加密进行处理！*