[04] Cookie 概念和基本使用

1、Cookie 是什么

Cookie，中文名称为“小型文本文件”或“小甜饼”，指某些网站为了辨别用户身份而储存在用户本地终端上的数据（通常经过加密）。

很多网站在你浏览后，会在你电脑中留下小小的档案，也就是我们说的 Cookie，以便你再次浏览时，网站会读取它上次给你留下的 Cookie 资料，如果有的话，就可以根据内容来判断使用者，送出特定的网页内容。

因为 HTTP 协议是无状态的，即服务器不知道用户上一次做了什么，这严重阻碍了交互式 Web 应用程序的实现。所以 Cookie 就是用来绕开 HTTP 的无状态性的“额外手段”之一。

Cookie 的一个典型的应用是当登录一个网站时，网站往往会请求用户输入用户名和密码，并且用户可以勾选“下次自动登录”。如果勾选了，那么下次访问同一网站时，用户会发现没输入用户名和密码就已经登录了。这正是因为前一次登录时，服务器发送了包含登录凭据（用户名加密码的某种加密形式）的 Cookie 到用户的硬盘上。第二次登录时，（如果该 Cookie 尚未到期）浏览器会发送该 Cookie，服务器验证凭据，于是不必输入用户名和密码就让用户登录了。

所以：

- Cookie 是保存在客户端的小文本
- 保存位置分两种
  - （1）保存在客户端浏览器所占内存中，关闭浏览器后 Cookie 也消失
  - （2）保存在客户端 PC 机的硬盘上，设置有效时间，超期后失效

但是要注意的是，Cookie 既然能把小文本保存在客户端，并在服务器和客户端之间进行传输，那么意味着它也容易造成信息泄露，因此：

- 不用 Cookie 保存对保密性要求高的信息，如银行卡密码等
- 不用 Cookie 实现某些必要功能，防止 Cookie 删除后功能出现错误
- 可以通过浏览器设置阻止 Cookie，或手工清除 Cookie
- Cookie 放在请求头 Header 里，而不是主体 Body 中，所以 GET 或 POST 方式的请求都可以发送 Cookie

另外，Cookie 还有一些缺陷：

- Cookie 会被附加在每个 HTTP 请求中，所以无形中增加了流量
- 由于在 HTTP 请求中的 Cookie 是明文传递的，所以安全性成问题（除非用 HTTPS）
- Cookie 的大小限制在 4KB 左右。对于复杂的存储需求来说是不够用的

如果你想看到自己的 Cookie，也打开某个网站，在控制台调用 JS 代码： javascript:alert("Cookie:"+document.cookie)
或 javascript:document.write(document.cookie)

2、Java 中的 Cookie 类

在 Servlet API 中，存在 Cookie 类，可以使用 new 关键字进行创建：

```
1. Cookie cookie = new Cookie("username", "zhangsan");
```

可以看到，Cookie 对象是保存一对键值对，都是字符串形式。

Cookie 类定义了一系列的方法，摘要部分如下：

类型	方法名	说明
void	setMaxAge(int expiry)	设置 Cookie 有效期，以秒为单位，保存在硬盘上或内存中 • 正数：保存到客户端硬盘上，一定时间有效 • 负数：浏览器关闭时 Cookie 被删除 • 零：Cookie 被删除
void	setValue(String value)	Cookie 创建后，对 Cookie 进行赋值
String	getName()	获取 Cookie 的名称
String	getValue()	获取 Cookie 的值
String	getMaxAge()	获取 Cookie 的有效时间，以秒为单位

保存和获取 Cookie：

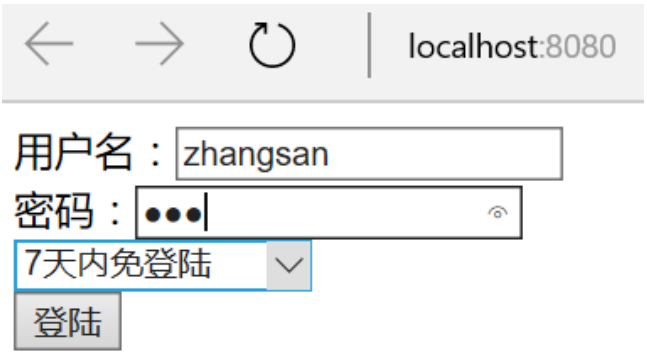
- 把 Cookie 保存到客户端 response.addCookie(Cookie cookie)
- 获取请求中的 Cookies request.getCookies()

Cookie 要保存到客户端，凡是写到客户端的方法，基本都是在响应中，HttpServletResponse 提供了方法，把 Cookie 保存到客户端；而再次访问与保存 Cookie 相同域名的网站时，HTTP 协议将把有效时间内的 Cookie 都发送到服务器，容器将 Cookie 封装到请求中，HttpServletRequest 提供了获取 Cookie 对象数组的方法。

3、一个示例带你看 Cookie

上面单纯文字的说明太过干瘪，那么就来一个生动鲜活的例子：

写登陆页面展示如下：




```jsp
<%@ page language="java" contentType="text/html; charset=utf-8"
pageEncoding="utf-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>登陆页面</title>
</head>
<body>
<%
String username = null;
String password = null;

//获取请求中的所有 cookie 对象
Cookie[] cookies = request.getCookies();

//只要存在 Cookie，就查找是否有用户名和密码
if(cookies != null) {
        for(Cookie cookiesTemp : cookies) {
                if(cookiesTemp.getName().equals("username")) {
                        username = cookiesTemp.getValue();
                }
                if(cookiesTemp.getName().equals("password")) {
                        password = cookiesTemp.getValue();
                }
        }
}

if(username != null && password != null) {
        request.getRequestDispatcher("LoginServlet?username=" +
username + "&pwd=" + password).forward(request, response);
}
%>

```

```
<form name="form1" action="LoginServlet" method="post">
用户名：<input type="text" id="username" name="username"><br>
密码：<input type="password" id="pwd" name="pwd"><br>
<select id="timelength" name="timelength">
        <option value="0" selected>每次都需要登陆</option>
        <option value="3">3 天内免登陆</option>
        <option value="7">7 天内免登陆</option>
</select><br>
<input type="submit" name="submit" value="登陆">
</form>

</body>
</html>
```

写后台 Servlet 代码如下：

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html; charset = utf-8");
    //获取用户名、密码
    String username = request.getParameter("username");
    String password = request.getParameter("pwd");

    //获得 JSP 页面的时间信息
    String timelength = request.getParameter("timelength");
    int days = 0;

    //类型转换
    if(timelength != null) {
        days = Integer.parseInt(timelength);
    }
    //只要天数不为 0，则创建 cookie，设置有效时间，存到客户端
    if(days != 0) {
        Cookie usernameCookie = new Cookie("username", username);
        Cookie passwordCookie = new Cookie("password", password);
        usernameCookie.setMaxAge(days * 24 * 3600);
        passwordCookie.setMaxAge(days * 24 * 3600);
        response.addCookie(usernameCookie);
        response.addCookie(passwordCookie);
    }

    //跳转到 home.jsp
    request.getRequestDispatcher("home.jsp").forward(request, response);
}
```

该示例中，输入账户密码登陆后，Servlet 中最后会调用 response.addCookie 方法，将账号密码存到客户端的硬盘上。

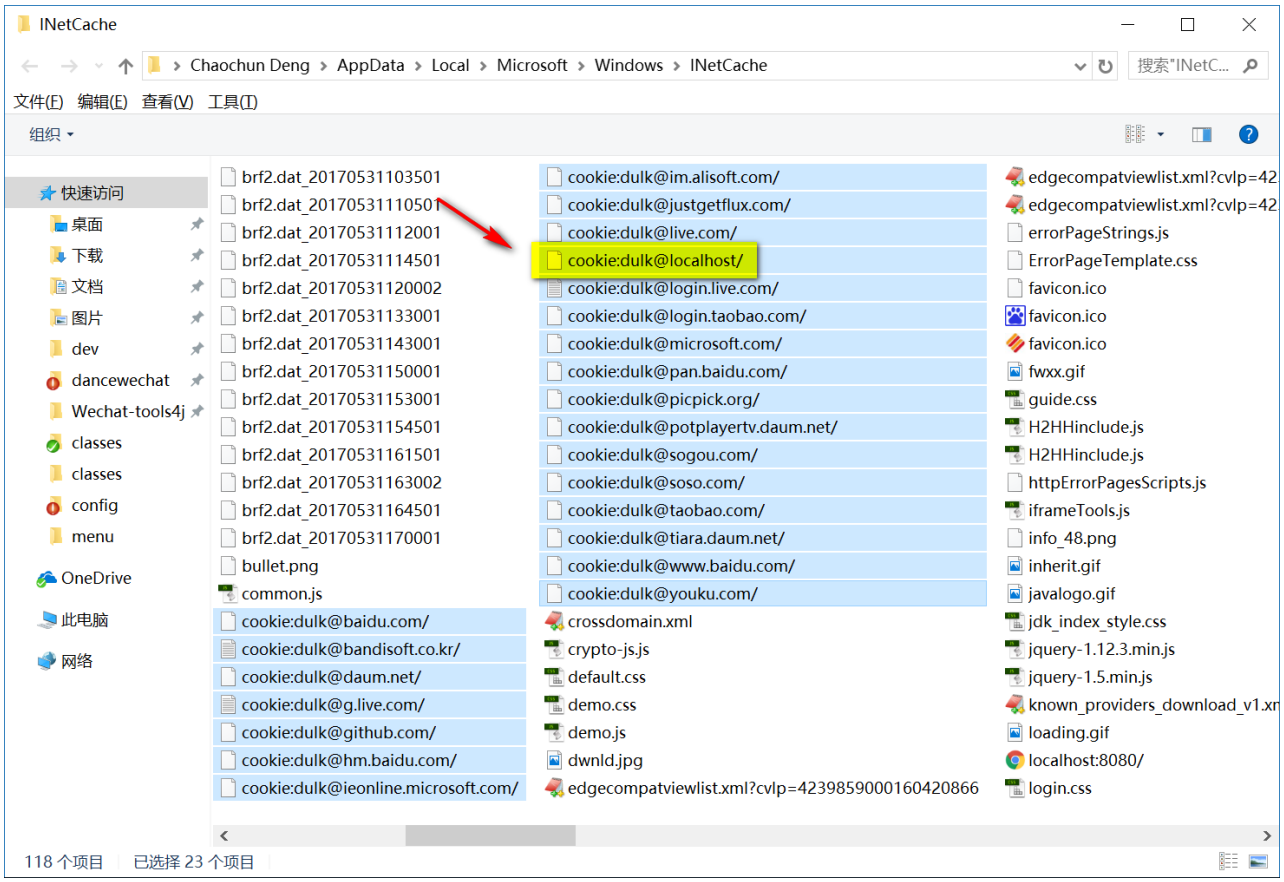
我们就到硬盘里去翻一翻，在这之前要说明的是：

- 不同浏览器的 Cookie 存放位置不同
- 不同浏览器的 Cookie 存放文件格式不同，IE 采用明文的文本文件，Safari 采用二进制文件，Chrome 和 Firefox 采用的是 Sqlite 数据库文件格式

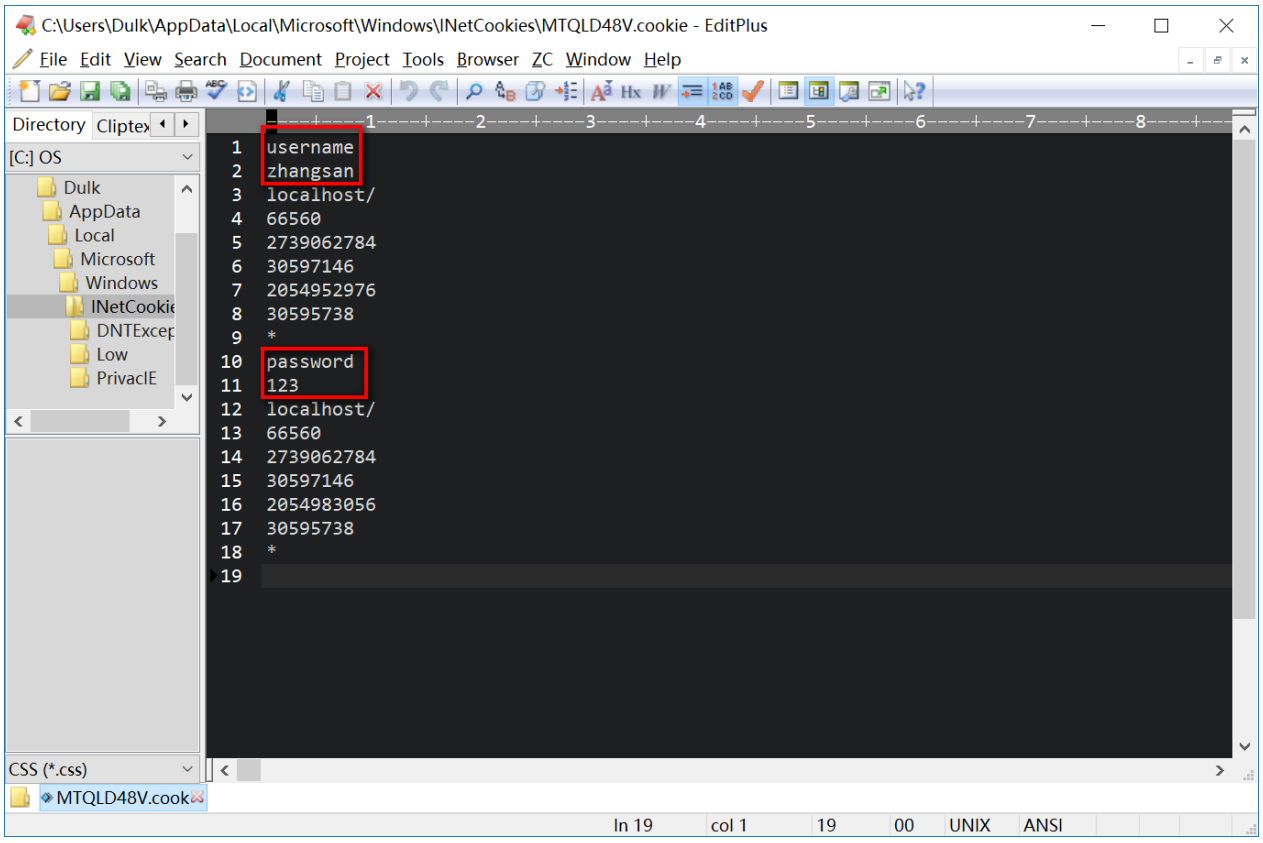
3.1 IE

为了方便展示，我们用 IE 浏览器登陆一次，然后可以在浏览器设置中，找到存放 Cookie 文件的位置：

可以看到，除了刚才登陆的 localhost，也还有平时登陆其他网站留下的 Cookie，我们看刚才登陆的 localhost 的 Cookie 就行了：

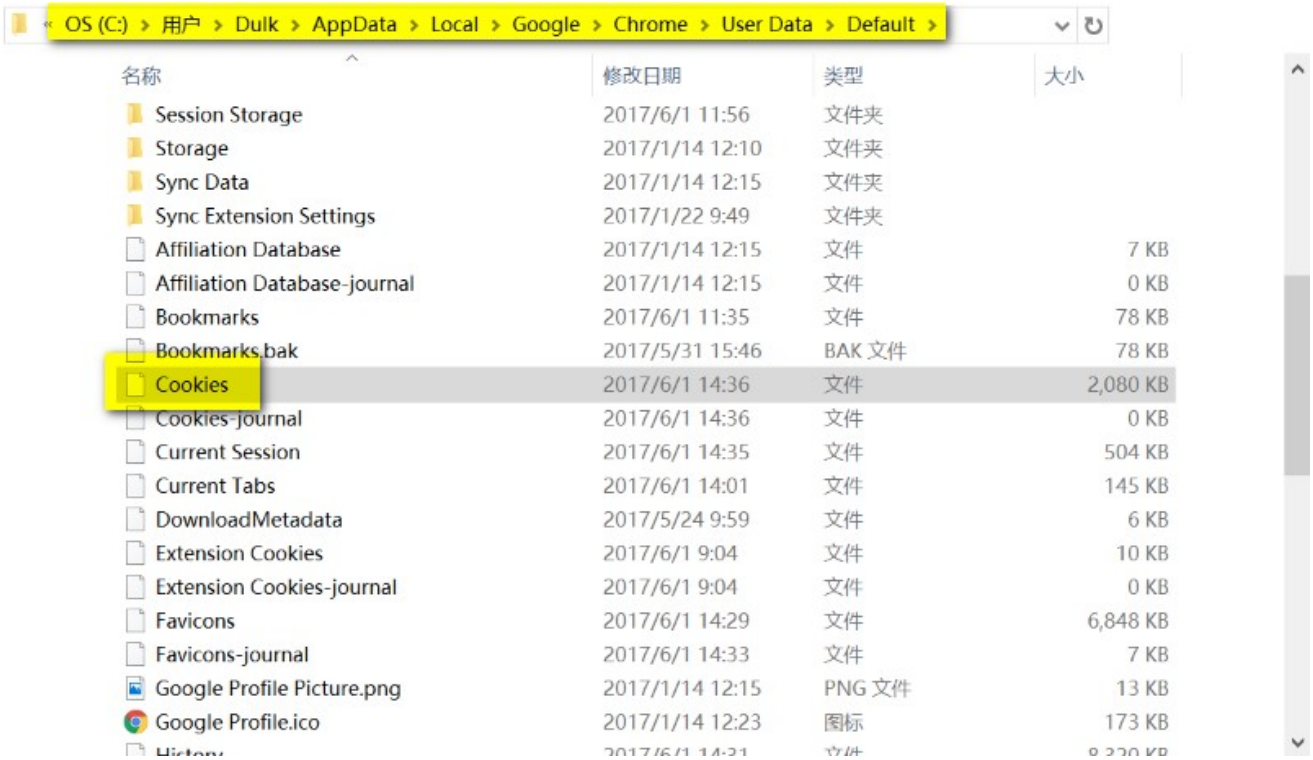


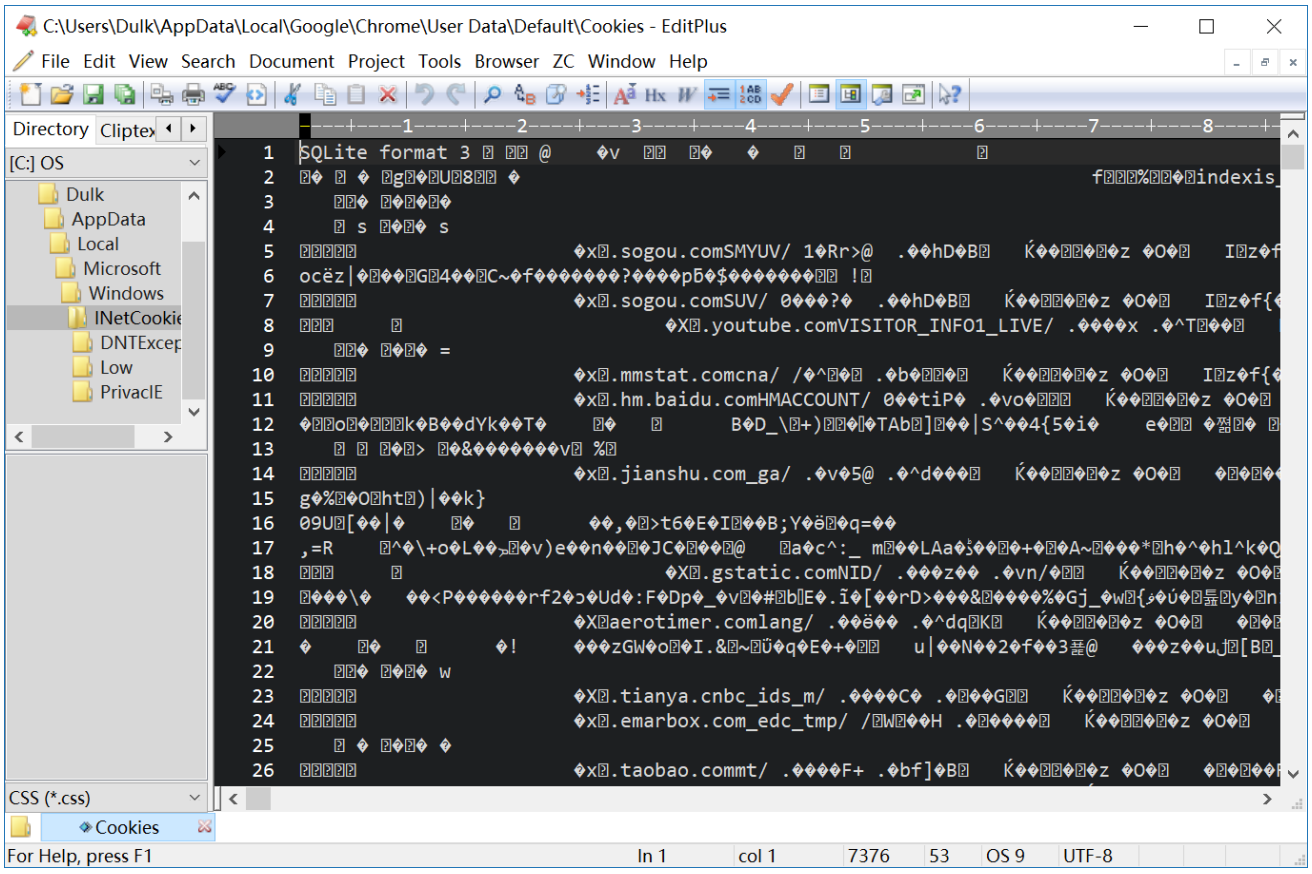
可见，确实把我们的账号和密码放到了 Cookie 中，并发送到了客户端的硬盘里：



3.2 Chrome

Chrome 因为文件格式问题，直接在硬盘上打开实际上你看到的基本都是乱码：





但索性我们还是可以直接在浏览器中看到的：

4、Cookie 的格式

我们已经知道了 Cookie 是个小文本，而且也基本是按照 key/value 的方式存储，所以也才有了封装的 Cookie 对象用来保存一对键值对。

Cookie 的内容当然不止我们定义的那对 key/value，实际上它是有一定格式的。Cookie 的标准格式定义在不同浏览器中实现，在历史上也有差异。下面是常用的标准格式，或者说属性：

- name　用户可定义的 key
- value　用户可定义的 value
- comment　用来提供注释说明
- path　指定 Cookie 作用路径，和 Domain 配合限制 Cookie 的作用范围
- domain　指定作用域
- max-age　指定 Cookie 失效时间
- secure　用来远程发送 Cookie 时告知浏览器数据已加密，只能 HTTPS 连接被发送
- httponly　本地 JavaScript 无法读取 Cookie 信息

这里再说明一下 path 和 domain，假设写 Cookie 的程序的访问路径是 http://localhost:8080/JavaWeb/servlet/CookieDemo

- localhost 就是域名
- /JavaWeb/servlet 就是当前 Cookie 的 path

假设现浏览器存的 cookie 的路径是/JavaWeb/servlet/，则：

- 访问的地址是：http://localhost:8080/JavaWeb/servlet/CookieDemo　则带该 cookie
- 访问的地址是：http://localhost:8080/JavaWeb/CookieDemo　则不带该 cookie

也就是说，Cookie 有个不可跨域名性，就像访问 Google 只会带上 Google 的 Cookie，而不会带上 Baidu 的 Cookie。

如下示例图，访问淘宝网时携带的 Cookie：

5、最后

因为 Cookie 实现自动登录涉及到用户信息安全的问题，实际上真正的自动登录不会像上面这么简单粗暴的，从我们查看 Cookie 就可以知道，账号密码都可以赤裸裸地看到，这显然是不行的。

但是这里不做展开，留下部分参考，以便未来学习：

- 理解 Cookie 和 Session 机制
- 记住密码功能如何设计？
- 网站的下次自动登录功能的实现方法
- cookie 实现自动登录