彻底理解cookie、session、localStorage、sessionStorage以及使用方法。

1. Cookie

1.1、cookie的由来

Cookie是1993年由网景公司前雇员发明的一种进行网络会话状态跟踪的技术。

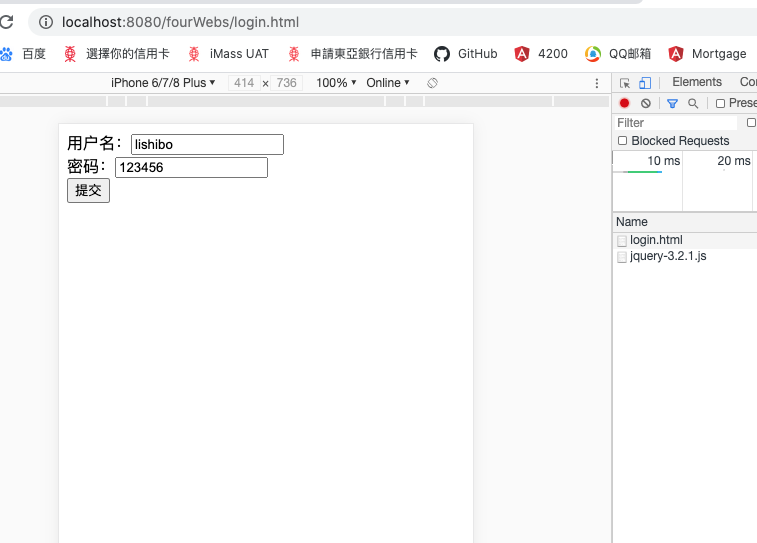
会话(对于客户端来说，会话可以理解为打开浏览器进入某个网站到关闭浏览器退出某个网站。)是由一组请求与响应组成，是围绕着一件相关事情所进行的请求与响应。所以这些请求与响应之间一定是需要有数据传递的，即是需要进行会话状态跟踪的。然而HTTP协议是一种无状态协议，在不同的请求间是无法进行数据传递的。此时就需要一种可以进行请求间数据传递的会话跟踪技术，而cookie就是一种这样的技术。

1.2、cookie的使用

a、cookie是由服务器生成，保存在客户端的一种信息载体。用户在提交第一次请求后，由服务器生成Cookie，并将其封装到响应头中，以响应的形式发送给客户端，客户端接收到这个响应后，将Cookie保存在客户端，当客户端再次发送同类请求时，在请求中会携带保存在客户端的Cookie数据，发送到服务端，由服务器对会话进行跟踪。

如下是一个登录页面，点击提交，call后端接口doLogin，后端生成Cookie并将Cookie添加到响应头中。当后续访问index页面时，浏览器会自动携带Cookie信息发送给服务器端。

前端login页面



**Java doLogin 代码**

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

System.***out***.print("doLogin");

//接收从客户端提交到服务器端的userName和passWord

String userName = request.getParameter("userName");

String passWord = request.getParameter("passWord");

//创建两个Cookie，用来存储userName和passWord

Cookie cookie1 = **new** Cookie("userName", userName);

Cookie cookie2 = **new** Cookie("passWord", passWord);

//指定Cookie绑定的路径

cookie1.setPath("/");

cookie2.setPath("/");

//向响应头中添加Cookie

response.addCookie(cookie1);

response.addCookie(cookie2);

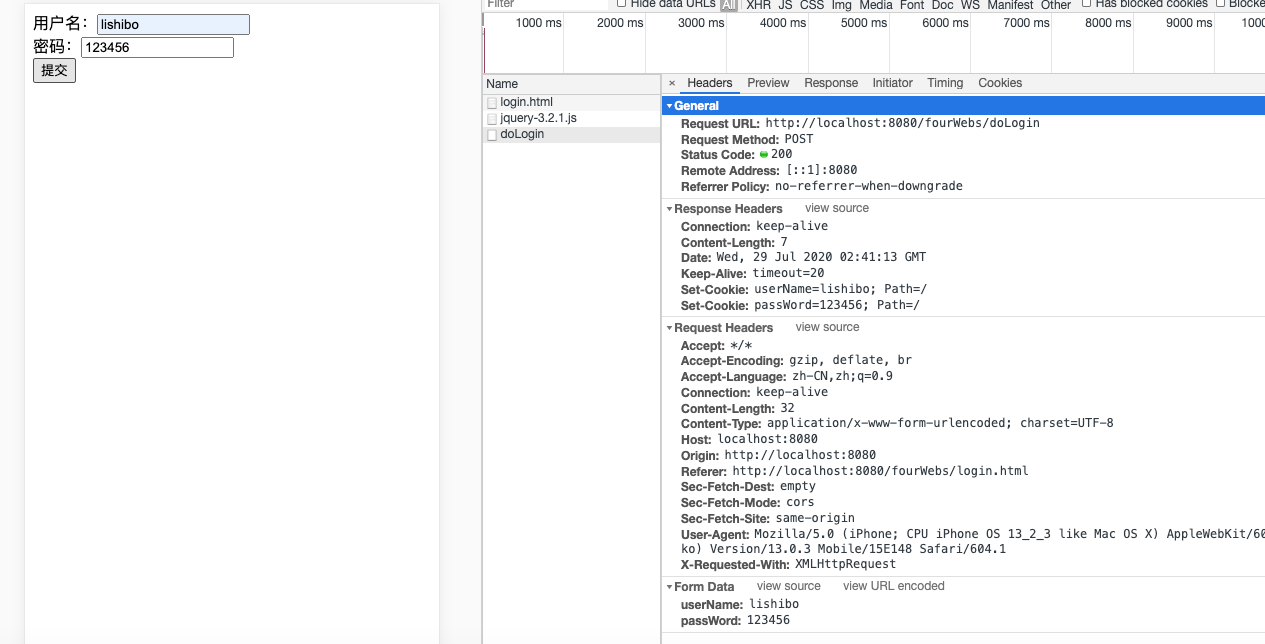
PrintWriter out = response.getWriter();

out.write("success");

out.close();

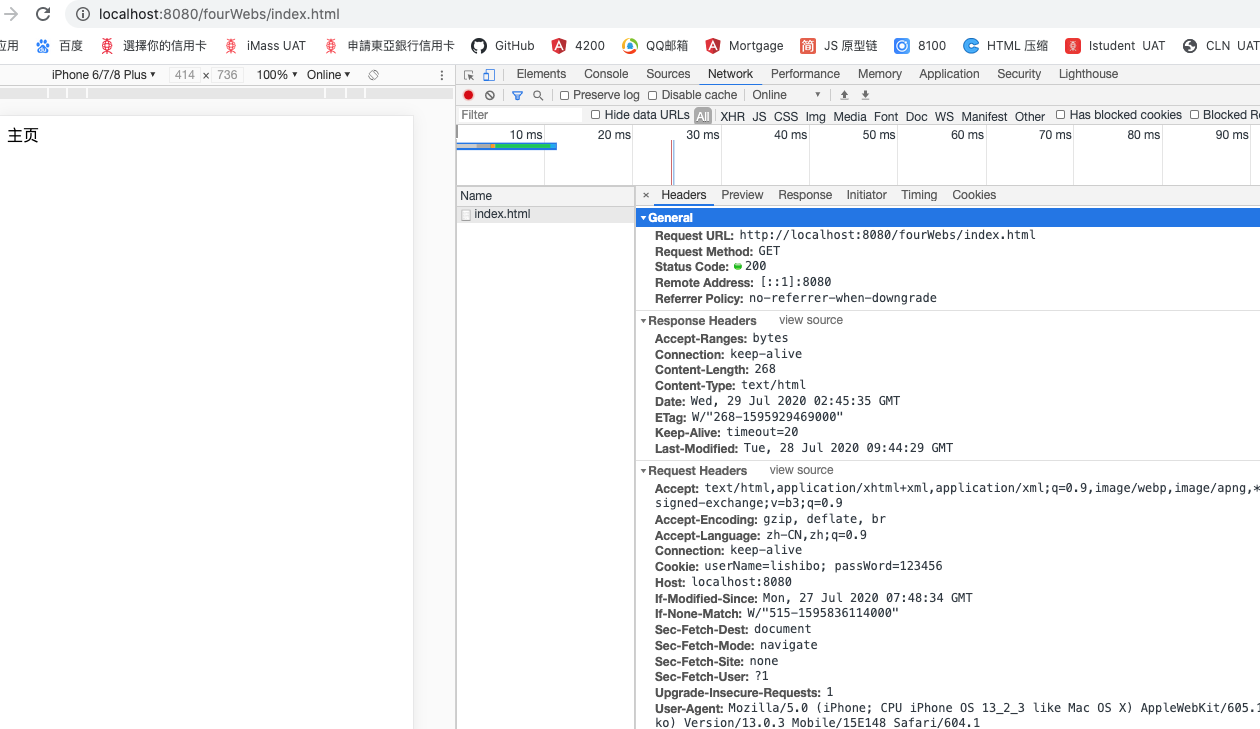
}

点击提交后，在doLogin接口的响应头中，我们可以看到已经将两个Cookie写入浏览器，如下



当再次访问该资源路径下的其它资源时，会将Cookie信息携带在请求头中，发送给服务器端，服务器端就可以获取Cookie处理相应的业务。

如访问index页面



1.3、cookie存储的地方。

Cookie分为持久性Cookie和会话级Cookie，当给Cookie设置过期时间时，Cookie信息就会被存储在计算机硬盘上，称为持久性Cookie。当Cookie没有设置过期时间时，Cookie信息就存储在浏览器内存中，当关闭浏览器时，Cookie就失效，称为会话级Cookie。上面我们演示的Cookie就是会话级Cookie，保存在浏览器内存中，当关掉浏览器再次打开浏览器访问index页面时，请求头中就不会携带Cookie信息。

不同的浏览器存储Cookie的方式不一样，因此存储的位置也会不同。比如使用Chrome浏览器打开某个网页，后端生成一个持久Cookie并将Cookie信息写入到响应头中，Cookie信息被保存在计算机的硬盘上，此时你在使用IE浏览器打开同一个网页，请求头中并不会携带Cookie信息，因为Chrome浏览器和IE浏览器存储Cookie的方式不一样，因此两个浏览器不能共用一个Cookie。对于会话级Cookie，Chrome浏览器是可以跨进程的，IE浏览器不能跨进程。

1.4、js操作Cookie。

/\*

设置Cookie

name 为Cookie的名称

value 为Cookie的值

day 为Cookie过期时间，单位为天

\*/

function setCookie(name, value, day) {

if (day !== 0) { //当设置的时间等于0时，不设置expires属性，cookie在浏览器关闭后失效。

var expires = day \* 24 \* 60 \* 60 \* 1000;

var date = new Date(+new Date() + expires);

document.cookie = name + "=" + escape(value) + ";expires=" + date.toUTCString();

} else {

document.cookie = name + "=" + escape(value);

}

}

/\*

获取Cookie

name为Cookie名称

\*/

function getCookie(name){

if(document.cookie.length != 0){

var cookieStart = document.cookie.indexOf(name + "=");

if(cookieStart != -1){

cookieStart = cookieStart + name.length + 1;

var cookieEnd = document.cookie.indexOf(";", cookieStart);

return cookieEnd != -1 ? document.cookie.slice(cookieStart, cookieEnd) : document.cookie.slice(cookieStart);

} else {

return "";

}

} else {

return "";

}

}

/\*

删除Cookie

name为Cookie名称

\*/

function delCookie(name){

// 由于Cookie不能直接删除，只需要将有效时间设置成当前时间之前的某个时间cookie就会失效。

setCookie(name, "", -1);

}

1. session

2.1、session介绍。

session是Web开发中的一种会话跟踪技术，前面说的Cookie也是一种会话跟踪技术，不同的是Cookie是将会话状态保存在客户端，而session则是将会话状态保存在服务器端。

2.2、session的使用。

还是使用上面login页面，当点击提交时call 后端接口CreateSession，在java程序中创建session并保存客户端发送过来的数据。

Java CreateSession代码

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

// 接收客户端发送过来的数据

String userName = request.getParameter("userName");

String passWord = request.getParameter("passWord");

//创建一个session对象，getSession()函数中不传参表示存在session对象就使用老session，不存在session对象就创建一个新的session对象

HttpSession session = request.getSession();

//向session域中写入属性

session.setAttribute("userName", userName);

session.setAttribute("passWord", passWord);

PrintWriter out = response.getWriter();

out.write("success");

out.close();

}

在java程序UsageSession中使用session

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

//getSession()函数参数是false时，表示有session对象就使用老session对象，没有不创建新的session对象

HttpSession session = request.getSession(**false**);

**if**(session != **null**) {

String userName = (String)session.getAttribute("userName");

String passWord = (String)session.getAttribute("passWord");

System.***out***.println(userName);

System.***out***.println(passWord);

}

}

那么问题来了，Cookie是储存在客户端用来标识客户信息的，session是存储在服务器端用来标识客户状态的，不同的客户使用自己的电脑访问同一个服务器，如果客户端没有存储Cookie信息，那在请求头中就不会携带Cookie信息，这样就能区分开不同的用户，但是当不同的用户都访问上面这个UsageSession服务时，服务端怎么能拿到正确的userName，而不会拿到其它用户的userName了？下面我们就来讲解一下session的工作原理。

2.3、session的工作原理。

1. 服务器对当前应用中的session是以Map的形式进行管理的，这个Map称为session列表。该Map的key为一个32为长度的随机串，这个随机串称为JSessionID，value则为session对象的引用。当用户第一次提交请求时，服务器中会执行request.getSession()方法，会自动生成一个Map.Entry对象，key为一个根据某种算法新生成的JSessionID，value则为新创建的session。
2. 在将session信息写入session列表后，系统还会自动将JSESSIONID作为name，这个32位长度的随机串作为value，以Cookie的形式存放到响应头中，并随着响应，将该Cookie发送到客户端。
3. 客户端接收到这个Cookie后，会将其自动存放到浏览器缓存中(Cookie没有设置有效时长，因此为会话级Cookie)。当用户再次提交其他请求时，请求头中就会携带这个Cookie，一块发送到服务器端。
4. 当服务端调用request.getSession(false)时，系统会自动检测请求中是否存在JSESSIONID，如果存在，就会使用这个32位的随机串到服务端Map列表中去查找相应的session，并处理对应业务。
5. localStorage。

3.1、localStorage介绍。

localStorage是HTML5新增的特性，这个特性主要是用来本地存储的，解决了Cookie存储空间不足的问题(Cookie中每条Cookie的存储空间为4k)，localStorage存储的内存大小是5M左右。

localStorage是本地永久存储，数据被存放在本地硬盘中，关掉浏览器数据不会被清除，在同一个浏览器的不同tab页中，localStorage是共享的，但是不同的浏览器存储的localStorage是不共享的，也就是说使用Chrome浏览器存储的localStorage在IE中是不能使用的。

3.2、localStorage的用法。

localStorage只能用来存储字符串。

localStorage.setItem("name","lisi"); //存储name = “lisi”。

localStorage.getItem(“name”); // 获取name的值。

localStorage.removeItem(“name”); //将name移除。

当localStorage要存储对象时，需要先将对象转化为json字符串进行存储。

var user= {"name":"lisi"; “age”:”24”};

localStorage.setItem("user",JSON.stringify(user)); //JSON.stringify(user)是将user对象转化为json字符串。

var user = JSON.parse(localStorage.getItem("user")) //将取出的json字符串转化为对象。

1. sessionStorage。

4.1、sessionStorage介绍。

和localStorage一样，sessionStorage也是HTML5新增的特性，也是用来本地存储数据的，不同的是localStorage是本地永久存储，sessionStorage存储的数据只有在同一个会话中才能被访问，关闭浏览器数据就会被清除。sessionStorage存储的数据是不能跨进程的，也就是在同一个浏览器的不同tab页中，sessionStorage不是共享的。sessionStorage数据存储在浏览器内存中，因此关闭浏览器数据就会被清除。

4.2、sessionStorage的用法。

和localStorage的用法相似，sessionStorage也只能存储字符串。

sessionStorage.setItem("name","lisi"); //存储name = “lisi”。

sessionStorage.getItem(“name”); // 获取name的值。

sessionStorage.removeItem(“name”); //将name移除。

当sessionStorage要存储对象时

var user= {"name":"lisi"; “age”:”24”};

sessionStorage.setItem("user",JSON.stringify(user)); //JSON.stringify(user)是将user对象转化为json字符串。

var user = JSON.parse(sessionStorage.getItem("user")) //将取出的json字符串转化为对象。