## **JavaWeb--Cookie、Session**

### **什么是Cookie？**

cookie是服务器告诉浏览器保存键值对的一种技术 cookie的大小不能超过4kb 创建cookie需要设置key值和value值

#### **CookieUtil工具类**

public class CookieUtil {  
​  
    /\*\*  
     \* 用于查找指定name的cookie值  
     \* @param name  
     \* @param cookies  
     \* @return  
     \*/  
    public static Cookie findCookie(String name, Cookie[] cookies){  
        if (name == null || cookies == null || cookies.length==0){  
            return null;  
       }  
        for (Cookie cookie : cookies) {  
            if (name.equals(cookie.getName())){  
                return cookie;  
           }  
       }  
        return null;  
   }  
}

#### **Cookie的创建**

使用Cookie可以实现免用户名登录，首先获取到请求的参数，然后将参数存进Cookie中，然后通过Jsp的隐藏对象Cookie获取到存入的数据，在页面上回显用户名

public void createCookie(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       创建Cookie对象，并写好Cookie的名字key值和value值  
//           通过响应头的Set-Cookie告诉浏览器创建的cookie的名字和value值  
        Cookie cookie = new Cookie("cookie1", "value1");  
        Cookie cookie1 = new Cookie("cookie2", "value2");  
        Cookie cookie2 = new Cookie("cookie3", "value3");  
        Cookie cookie3 = new Cookie("cookie4", "value4");  
        Cookie changeCookie1 = new Cookie("cookie2", "changeValue2");  
//       调用resp的方法addCookie告诉浏览器保存cookie对象，必须要加  
        resp.addCookie(cookie);  
        resp.addCookie(cookie1);  
        resp.addCookie(cookie2);  
        resp.addCookie(cookie3);  
        resp.addCookie(changeCookie1);  
        resp.getWriter().write("cookie创建好了");  
   }

#### **Cookie的删除**

   public void deleteCookie(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       删除Cookie实际就是将setMaxAge设置成0-->>代表立马关闭这个cookie，默认值是-1(只在这一次会话中有效，即浏览器关闭就失效)  
//       也可以自己设置它的寿命，setMaxAge中的数字代表秒数  
//       同样要在设置完setMaxAge后调用addCookies使修改生效  
        Cookie cookie = CookieUtil.findCookie("cookie4",req.getCookies());  
        if (cookie!=null){  
            cookie.setMaxAge(0);  
            resp.addCookie(cookie);  
       }  
        resp.getWriter().write("删除成功");  
   }

#### **Cookie的修改**

   public void updateCookie(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       第一种：创建一个cookie对象然后取和想更改的cookie对象一样的key值，对value进行更改  
        Cookie changeCookie = new Cookie("cookie3","aaaaa");  
//       通过响应头中的Set-Cookie告诉浏览器修改的对象和内容  
        resp.addCookie(changeCookie);  
        resp.getWriter().write("修改成功");  
   }

#### **Cookie的获取**

public void getCookie(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       通过请求头中的Cookie传递获取的cookie 的 key和value  
        Cookie[] cookies = req.getCookies();  
        Cookie iWantCookie = CookieUtil.findCookie("cookie1",cookies);  
        if (cookies.length == 0){  
            resp.getWriter().write("并没有cookies");  
       }  
        for (Cookie cookie : cookies) {  
            resp.getWriter().write("获取到了cookies的值");  
            resp.getWriter().write(cookie.getName()+"="+cookie.getValue()+"<br>");  
       }  
        if (iWantCookie!=null){  
            resp.getWriter().write("找到了");  
       }  
   }

#### **Cookie的Path属性设置**

使用setPath()可以设置Cookie的Path属性，后面要记得加addCookie()方法

public void setCookiePath(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       设置Path属性可以规定cookie在指定路径后的页面生成  
        Cookie cookie = new Cookie("PathCookie","Path");  
        cookie.setPath(req.getContextPath()+"/abc");  
        resp.addCookie(cookie);  
//       resp.getWriter().write("！！！！！！！！！！");  
   }

#### **Cookie的存活时间**

setMaxAge()方法能够设置Cookie的存活时间按，在默认情况下，Cookie只在一次会话中有效，默认值是-1，当设置为0就会立即关闭对应的cookie对象，设置为其他值就会在事件过后被销毁，单位是秒

public void setCookieLife(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
    Cookie cookie = new Cookie("defaultLife","defaultLife");  
    Cookie cookie1 = new Cookie("LongLife","LongLife");  
    cookie.setMaxAge(-1);  
    cookie1.setMaxAge(60 \* 60);  
    resp.addCookie(cookie);  
    resp.addCookie(cookie1);  
}

##### **Cookie的调用验证**

<div >  
<iframe name="cookie\_show" width="500" height="200"></iframe>  
​  
<ul>  
    <li><a href="cookieServlet?action=createCookie" target="cookie\_show">cookie的创建</a></li>  
    <li><a href="cookieServlet?action=setCookiePath" target="cookie\_show">cookie的Path属性的设置</a></li>  
    <li><a href="cookieServlet?action=getCookie" target="cookie\_show">cookie的获取</a></li>  
    <li><a href="" target="cookie\_show">cookie使用--免用户名登录(也能做到免密码登录)</a></li>  
    <li><a href="cookieServlet?action=updateCookie" target="cookie\_show">cookie的修改</a></li>  
    <li><a href="cookieServlet?action=deleteCookie" target="cookie\_show">cookie的删除</a></li>  
    <li><a href="cookieServlet?action=setCookieLife" target="cookie\_show">cookie的存活时间修改为1小时</a></li>  
</ul>  
</div>

### **什么是Session？**

Session是一个接口，是一个会话，他是维护客户端和服务器之间关联的技术 每个客户端（浏览器）中都有一个Session 经常用Session保存用户登陆后的信息

#### **创建Session和验证Session是不是新创建的**

   public void createSession(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       创建Session--->>>req.getSession，这一句就行  
        HttpSession session = req.getSession();  
//       在没有Session之前就会创建一个新的Session，有就会用之前的Session  
        resp.getWriter().write("创建了Session <br>");  
//       isNew就是判断Session对象是不是新创建的  
        boolean aNew = session.isNew();  
        resp.getWriter().write("是不是新创建的Session: " + aNew + "<br>");  
//       获取Session的ID--->>>Session.getId()  
        resp.getWriter().write("id: <br>" + session.getId());  
   }

#### **Session中存取数据**

   public void setSession(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       先调用这个方法就会重新创建一个Session对象  
        req.getSession().setAttribute("key1", "value1");  
        resp.getWriter().write("存入了数据");  
   }  
​  
    public void getSession(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
        Object key1 = req.getSession().getAttribute("key1");  
        resp.getWriter().write("Session域中的对象的value值为:" + key1);  
   }

#### **Session的生命周期**

##### **session的最大请求间隔时长**

getMaxInactiveInterval获取到的时间是Session默认的存在时间，单位是秒

   public void getSessionTime(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       getMaxInactiveInterval会获得Session对象的超时时间，setMaxInactiveInterval能设置超时时间，过了时间就会被销毁  
/\*       在web.xml文件中可以直接通过<session-config>  
                                    <session-timeout></session-timeout>  
                                </session-config> 标签对规定在工程下创建的所有Session的默认超时时间\*/  
        int maxInactiveInterval = req.getSession().getMaxInactiveInterval();  
        resp.getWriter().write("Session的默认超时时间为:" + maxInactiveInterval);  
   }

##### **设置Session默认存活时间**

<session-config>  
    <session-timeout>默认存活时间</session-timeout>  
</session-config>

#### **设置最大请求间隔时长**

MaxInactiveInterval所设置的是每次请求间隔的最大时间，当点击请求之间的间隔时间超过所设置的值Session就会被销毁

   public void setSessionTime(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       setMaxInactiveInterval设置Session对象的请求间隔最大时间，超过时间就会销毁Session对象，在调用就会创建一个新的  
//       设置为负数永不过期，很少使用  
        req.getSession().setMaxInactiveInterval(3);  
        resp.getWriter().write("是不是新创建的Session:   "+req.getSession().isNew());  
   }

##### **销毁Session对象**

   public void deleteSession(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
//       invalidate这个方法会直接让当前Session对象失效（过期）  
        req.getSession().invalidate();  
        resp.getWriter().write("Session对象被删除了");  
   }

#### **Seeion和浏览器、Cookie的关系**

在将服务器启动后点击创建Session的按钮，会创建一个新的Session对象，在这时会发现浏览器也创建了一个Cookie对象，将新创建的Session的id保存到服务器内存中，当再次发起请求查找Session对象时，浏览器会根据请求头中的Cookie: JSESSIONID=48A6A12B9718446164E98F6224A655E3;值在服务器中查找Sessionid对应的Session对象，然后将结果返回至浏览器。

当在浏览器的控制台删除了Cookie对象后，再次进行请求查找Session对象就会新创建一个新的Session对象，因为并没有从请求头中接收到Cookie保存的Session id，无法进行查找，就会创建一个新的Session对象，重新将ID保存在Cookie中，多次请求依旧能够发现Session并没有创建新的。