一、会话技术-cookie

2019年2月10日 22:05

1. 什么是会话?

   会话: 为了实现某一个功能, 客户端和服务器之间可能会产生多次的请求和响应, 从客户端访问服务器开始, 到最后访问服务器结束, 客户端关闭为止, 这期间产生的多次请求和响应加在一起就称之为是客户端和服务器之间的一次会话

2. 如何来保存会话过程中产生的数据?

   ~requset域太小

   ~ServletContext域太大

3. Cookie

   a. Cookie概述

   Cookie的原理是通过Set-Cookie响应头和Cookie请求头将会话中产生的数据保存在客户端

   i. 客户端请求服务器, 服务器将需要保存的数据通过Set-Cookie响应头发给客户端, 客户端收到后会将数据保存在浏览器的内部。

   ii. 当客户端再次请求服务器时, 通过Cookie请求头将上次保存的数据再带给服务器, 服务器通过Cookie头来获取数据, 通过这种方式可以保存会话中产生的数据。

   iii. Cookie是将需要保存的数据保存在了客户端, 是客户端技术. 每个客户端各自保存各自的数据, 再次访问服务器时会带着自己的数据, 每个客户端持有自己的数据, 因此就不会发生混乱了。

   b. 实现Cookie

   SUN公司为了简化Cookie的开发, 提供了如下操作Cookie的API:

   i. 创建Cookie

      Cookie cookie = new Cookie(String name, String value);

      getName()

      getvalue()

      setValue()

   ii. 添加Cookie

      response.addCookie();

         //向响应中添加一个Cookie, 可以在一次响应中添加多个Cookie

   iii. 获取Cookie

      Cookie [] cs = requset.getCookies();

         //返回请求中所有Cookie组成的数组, 如果请求中没有任何Cookie信息, 则返回null.

   iv. 设置Cookie存活时间

      setMaxAge();//指定Cookie保存的时间, 以秒为单位

         //如果不明确的指定, Cookie默认是会话级别的Cookie, Cookie会保存在浏览器的内存中, 一旦浏览器关闭, Cookie也会随着浏览器内存的释放而销毁.

         //通过setMaxAge()方法可以设置Cookie的存活时间, 设置了存活时间后, Cookie将会以文件的形式保存在浏览器的临时文件夹中, 在指定的时间到来之前, 即使多次开关浏览器, Cookie信息也会一直存在.

   v. 设置Cookie路径

      setPath(String path);

         //设置当前Cookie在浏览器访问哪一个路径及其子孙路径的时候带回来

         //如果不指定, 默认的path值就是发送Cookie的Servlet的所在的路径

   vi. 删除Cookie

没有直接删除Cookie的方法!!!

如果想要删除一个Cookie, 可以向浏览器发送一个 同名 同path 的Cookie, 只需要将Cookie的maxAge设置为0, 由于浏览器是根据 名+path+domain 来区分Cookie的, 所以当两个cookie的上述条件相同时, 浏览器就会认为是同一个Cookie, 那么后发的Cookie会覆盖之前的, 而后发的Cookie的存活时间为0, 所以浏览器收到后也会立即删除!!

vii. ~Cookie的细节:

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）。

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。

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4KB。

4. 案例: 在页面中显示上次访问时间

a. 第一次实现CookieDemo1:

```
//解决乱码
    response.setContentType("text/html;charset=utf-8");
    String dateStr = new Date().toLocaleString();
    response.setHeader("Set-Cookie", "time="+dateStr);
//获取上次访问的时间
    String dateStr2 = request.getHeader("cookie");
    response.getWriter().write("上次访问时间为："+dateStr2);
```

b. 利用Cookie对象实现CookieDemo2:

```
    //乱码
        response.setContentType("text/html;charset=utf-8");
    //1.获取时间字符串
        String dateStr1 =new Date().toLocaleString();
    //2.创建Cookie
        Cookie cookie = new Cookie("time",dateStr1);
    //设置cookie生存时间
        cookie.setMaxAge(3600);//单位是秒
//      cookie.setPath("/day12");
        cookie.setPath(request.getContextPath()+"/");
    //3.发送Cookie
        response.addCookie(cookie);
    //4.获取上次访问时间
    //5.获取所有Cookie
        Cookie timeC = null;
        Cookie[] cs = request.getCookies();
        if(cs != null){
            for (Cookie c : cs) {
                if("time".equals(c.getName())){
                    timeC = c;
                }
            }
        }
    //6.打印时间
```

```
if(timeC != null){
    response.getWriter().write("上次访问时间："+timeC.getValue());
}else{
    response.getWriter().write("您是第一次访问本网站！");
}
```

5. 案例: EasyMall登陆功能之记住用户名
   a. 引入login.html页面，将其修改为login.jsp文件
   b. 修改_head.jsp中的登录标签href属性为如下内容：
      i. `<a href="<%=request.getContextPath() %>/login.jsp">登录</a>`
   c. 编写LoginServlet, 代码实现如下:

```
//获取记住用户名cookie
String remname = request.getParameter("remname");
//记住用户名
if("true".equals(remname)){
    Cookie remnameCookie = new Cookie("remname", URLEncoder.encode(username, "utf-8"));
//设置最大生存时间(保存30天)
    remnameCookie.setMaxAge(3600*24*30);
//设置path, 让浏览器访问当前WEB应用下任何一个资源都能带Cookie!!
    remnameCookie.setPath(request.getContextPath()+"/");
//将Cookie添加到response中发送给浏览器
    response.addCookie(remnameCookie);

}else{//取消记住用户名 -- 删除Cookie
    Cookie remnameCookie = new Cookie("remname", "");
    remnameCookie.setMaxAge(0);
    remnameCookie.setPath(request.getContextPath()+"/");
    response.addCookie(remnameCookie);
}
```

   d. 在login.jsp的form表单中添加如下内容：

```
<form action="<%=request.getContextPath() %>/LoginServlet" method="POST">
```

   e. 编写login.jsp, 从Cookie中获取用户名存入用户名输入框

```
<%
//获取Cookie中的用户名
    Cookie[] cs = request.getCookies();
    Cookie remnameCookie = null;
    if(cs != null){
        for(Cookie c : cs){
            if("remname".equals(c.getName())){
                remnameCookie = c;
            }
        }
    }
    String username = "";
    if(remnameCookie != null){
```

```
            username = URLDecoder.decode(remnameCookie.getValue(), "utf-8");
        }
    %>
```

在input[name='username']中添加如下内容:

```
        <input type="text" name="username" value="<%=username%>"/>
```

f. 勾选记住用户名复选框

```
    <input type="checkbox" name="remname" value="true"
        <%= remnameCookie == null ? "" : "checked='checked'" %>
    />记住用户名
```

二、会话技术-Session

2019年2月10日 22:27

1. Session
   a. Session概述

      Session是将会话中产生的数据保存在了服务器端, 是服务器端技术

   b. 获取session对象：

      request.getSession();//如果有session直接拿过来使用，如果没有则创建一个

      request.getSession(true);//如果有session直接拿过来使用，如果没有则创建一个

      request.getSession(false);//如果有session直接拿过来使用,如果没有session则返回null

   c. Session作用：

      Session是一个域对象

      setAttribute(String name, Object value);

      getAttribute(String name);

      removeAttribute(String name)

      getAttributeNames()

      i. 生命周期:

         当第一次调用request.getSession()方法时创建Session。

         超时: 如果一个Session超时30分钟(可以在web.xml中来修改, 在根目录下通过<session-config>来配置)未被使用, 则认为Session超时, 销毁session

         自杀: 当调用session.invalidate()方法的时session立即销毁!!

         意外身亡: 当服务器非正常关闭时, 随着应用的销毁, session销毁. 当服务器正常关闭, 则未超时的session将会以文件的形式保存在tomcat服务器work目录下, 这个过程叫做session的钝化. 当服务器再次启动时, 钝化着的session还可以恢复过来, 这个过程叫做session的活化。

      ii. 作用范围: 当前会话范围

      iii. 主要功能: 保存当前会话相关的数据

   d. Session的原理

      Session是基于一个JSESSIOINID的Cookie工作的

      i. 怎么解决浏览器关闭之后, 无法使用浏览器关闭之前的Session??

         (注意: 在访问一个jsp时, 默认一上来就会创建session!!)

         Cookie cookie = new Cookie("JSESSIONID",session.getId());

         cookie.setPath(request.getContextPath()+"/");

         cookie.setMaxAge(30*60);

   e. 案例: 实现购物车

      i. 修改index.jsp页面为sale.jsp，并添加内容如下：

         1) 页面第一行添加： session="false"

         2) <a href="<%=request.getContextPath()%>/servlet/BuyServlet?pord=小米手机">小米手机

         <a href="<%=request.getContextPath()%>/servlet/BuyServlet?pord=海尔洗衣机">海尔洗衣机

         <a href="<%=request.getContextPath()%>/servlet/BuyServlet?pord=拖鞋">拖鞋
         <a href="<%=request.getContextPath()%>/servlet/PayServlet">支付

      ii. 创建BuyServlet和PayServlet：

      BuyServlet：

      //1.获取商品信息

      String prod = request.getParameter("pord");

      prod = new String(prod.getBytes("iso8859-1"),"utf-8");//页面中是get请求

```
//2.获取session对象
    HttpSession session = request.getSession();//如果有session直接拿过来使用，如果没有则创建一个
    session.setAttribute("prod", prod);
    response.setContentType("text/html;charset=utf-8");
    Cookie cookie = new Cookie("JSESSIONID",session.getId());
    cookie.setPath(request.getContextPath()+"/");
    cookie.setMaxAge(30*60);        -----怎么解决浏览器关闭之后,无法使用浏览器关闭之前的Session??
    response.addCookie(cookie);    -----获取session，将其保存30分钟。
//3.做出回应
    response.getWriter().write("恭喜，将["+prod+"]加入购物车");
```

PayServlet：

```
    response.setContentType("text/html;charset=utf-8");
    HttpSession session = request.getSession(false);
    if(session != null){  //session可能为空
        String prod = (String) session.getAttribute("prod");
        response.getWriter().write("成功为"+prod+"结算");

    }else{
        response.getWriter().write("没选商品");
    }
```

f. EasyMall案例: Easymall登陆功能的实现

所谓的登陆,就是在检查用户提交的用户名和密码之后,对于通过校验的用户,在session中保存一个登陆标记,从而在访问其他页面的时候,可以通过在session中获取用户的登陆标记,来判断当前用户的登陆状态

i. 导入登陆页面

将登陆页面及相关的css及图片拷贝到工程中. 用jsp代替html展示数据

ii. 编辑login.jsp页面

(1) 修改首页头部中的登陆超链接,将url地址指向login.jsp

(2) 修改login.jsp中的url地址,将相对路径改为绝对路径

(3) 修改表单提交地址,将表单提交到LoginServlet

iii. 实现后台注册代码

1) 创建LoginServlet,处理登陆请求

2) 编写LoginServlet,代码实现如下:

```
//1.处理乱码
request.setCharacterEncoding("utf-8");
//2.获取请求参数
String username = request.getParameter("username");
String password = request.getParameter("password");
String remname = request.getParameter("remname");
//3.记住用户名(见上)
//4.登陆用户
rs = ps.executeQuery();
if(rs.next()){
    //去登陆
    request.getSession().setAttribute("user", username);
    //登陆成功,跳转到首页
```

```
            response.sendRedirect(request.getContextPath()+"/index.jsp");
        }else{
            //用户名或密码不正确,转发回登陆页面提示
            request.setAttribute("msg", "用户名或密码不正确");
            request.getRequestDispatcher("/login.jsp").forward(request, response);
            return;
        }
```

3) 编写login.jsp,在登陆table中添加行获取登陆失败时的提示消息

```
<td class="tdx" colspan="2" style="color:red;text-align: center;">
    <%= request.getAttribute("msg") == null ? "" : request.getAttribute("msg") %>
</td>
```

g. EasyMall案例: 校验验证码

i. 编写ValiImageServlet,将验证码文本存入session

```
request.getSession().setAttribute("code", code);
```

ii. 在RegistServlet中,从session中取出验证码,和用户提交过来的进行比较

1) 将验证码的非空校验提到紧邻获取参数代码之后。

2) 获取session中的存储的验证码code:

```
String code = (String)request.getSession().getAttribute("code");
if(!valistr.equalsIgnoreCase(code)){
//设置提示消息
    request.setAttribute("msg", "验证码不正确");
//转发回注册页面进行提示
    request.getRequestDispatcher("/regist.jsp").forward(request, response);
return;
}
```

三、Cookie和Session的比较

2019年2月10日 22:36

1. Cookie和Session的比较:
   a. Cookie是将会话中产生的数据保存在客户端, 是客户端的技术
   b. Session是将会话中产生的数据保存在服务器端, 是服务器端的技术
   c. Cookie保存的信息的时间比较长, 但是安全性不高. 可能随着用户的操作, Cookie会被清空, 所以Cookie存储数据的稳定性比较差. 因此Cookie适合存放要保存时间较长, 但安全性要求不高的信息
   d. Session通常保存信息的时间比较有限, 但安全性比较高, 因为是保存在服务器端, 不会随着用户的操作而导致Session意外丢失, 因此session适合存放安全性要求比较高, 但是不需要长时间保存的数据.