会话技术 -- cookie

2019年5月21日 星期二 9:13

1. 会话是什么？

   为了完成某一个功能，浏览器和服务器建立连接之后，进行一次或多次请求和响应操作，这些请求和响应构成了浏览器和服务器之间的一次会话操作。

2. 会话的作用：

   会话主要是解决在浏览器请求中数据的共享问题而出现的技术。

   http是一个无状态协议。 --- 请求与请求之间是没有关联的。

   --- 利用会话技术可以解决请求之间没有关联的问题。

3. 会话技术--cookie

   a. Cookie的实现原理：

      在第一次请求过后，由服务器发出一个携带set-Cookie响应头的响应，到达浏览器之后，会将其中的数据在浏览端保存一份，以备下次发送请求时自动携带。在下一次发送请求时，请求中自动包含一个请求头cookie其中包含的值是set-cookie在浏览器端保存的值。

   b. 浏览器请求服务器，服务器响应的结果数据会保留在浏览器端一份，这个数据可以通过cookie技术来使用。所以cookie技术是一门浏览器端的技术。

4. 用户上次访问页面的时间

   在set-cookie中保存的时间是用户本次的访问时间，这个访问时间会在下一次请求中包含，并且通过请求头cookie获取，展示在浏览器端。

```
//在浏览器中回显用户上次访问页面的时间
public class CookieDemo1 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        //产生一个时间值
        Date date = new Date();
        String time = date.toLocaleString();
        //实现cookie。
        //1.设置setcookie响应头
        response.setHeader("set-cookie", time);
        //2.获取一个请求头cookie
        String cookie = request.getHeader("cookie");
        //如果用户是第一次访问当前页面，则在浏览器中没有cookie请求头，
        //所以无法获取到上次访问的时间
        if(cookie == null){
            response.getWriter().write("您是初次访问这个页面");

        }else{
            //如果不是初次访问，
            //则在cookie请求中会读取浏览器中保留的set-cookie响应头中上次访问时间
            response.getWriter().write("您上次访问本页面的时间是："+cookie);
```

```
        }
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);

    }

}
```

5. Cookie类型

Sun公司提供的Cookie类，便于开发者创建、使用cookie。

a. 创建cookie

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

b. 发送cookie到浏览器

```
response.addCookie(cookie);
```

c. 设置最大生命时长

```
cookie.setMaxAge();//单位为秒
```

如果不设置cookie 的最大生命时长，则cookie是一个会话级别的cookie，在浏览器关闭之后就会销毁，如果定义的了cookie 的生命时长，则cookie会以文件的形式保存在本地磁盘一份(浏览器的临时文件目录)，在到达执行生命时长之后cookie会自动失效，销毁.

d. 设置有效路径

```
cookie.setPath();
```

所谓有效路径就是cookie可以被访问的路径级别。

如果不指定cookie的有效路径级别，则只在当前资源的映射路径下有效。

如果指定cookie的有效路径，则在指定路径及其级别下都可以获取到当前cookie。

推荐在设置路径时加上一个"/"。如果web应用的虚拟路径为缺省路径，则此时将会以"/"作为有效路径，可以避免缺省路径带来的不必要的错误。

e. 获取cookie

```
Cookie[] cs = request.getCookies();
```

结果是一个cookie数据，将数组中的数据遍历使用。

f. 删除cookie

cookie中没有删除cookie的方法。

i. 创建一个名称要和删除的cookie名称相同的cookie，并且要设置这个cookie的path与要删除的相同，然后将新发送的cookie生命时长设置为0，这样就可以删除一个cookie。

ii. 原因：

浏览器在判断两个cookie是否相同时，判断三个内容：Name+Path+domain，默认情况下每一个cookie的domain都是相同的，所以只需要发送一个name和path与要删除的cookie相同的cookie到浏览器就可以完成删除操作了。

```
//创建一个name和path与要删除的完全一致的cookie，并且设置生命时长为0
    //创建cookie
```

```
Cookie cookie = new Cookie("time","");
//path一致
cookie.setPath(request.getContextPath()+"/");
//设置生命时长为0
cookie.setMaxAge(0);
//将cookie发送到浏览器
response.addCookie(cookie);
```

g. cookieAPI

cookie.getName()　----获取cookie 的名称

cookie. getValue()　　----- 获取cookie中包含的值

h. ~cookie小细节

一个cookie标识一种信息。

同一个cookie信息可以在多个浏览器中保存。

一个浏览器中可以保存不同网站的cookie。

6. Cookie类型使用cookie：

```
//cookie生命时长、有效路径设置
public class CookieDemo3 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        Date date = new Date();
        String time = date.toLocaleString();
        //创建cookie
        Cookie cookie = new Cookie("time",time);
        //修改cookie生命时长、有效路径
        //设置cookie生命时长（以秒值为单位）
        cookie.setMaxAge(60*60*24);
        //设置路径
        //修改为web应用的虚拟路径
        //如果是缺省web应用，则在request.getContextPath中获取到一个空的返回值。
        //这个值不能作为setPath的值，需要在其后加上一个"/"，来表明是当前web应用的路径。
        cookie.setPath(request.getContextPath()+"/");
        //发送cookie
        response.addCookie(cookie);
        Cookie timeC = null;
        //获取cookie
        Cookie[] cs = request.getCookies();
        if(cs != null){
            for(Cookie c:cs){
                if("time".equals(c.getName())){
                    timeC = c;
                }
            }
        }
        if(timeC != null){
            response.getWriter().write("您上次访问页面的时间为："+timeC.getValue());
        }else{
            response.getWriter().write("您是初次访问本页面");

        }
```

```
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);

    }

}
```

7. 修改EasyMall ： 添加登录功能--记住用户名
   a. 添加login.jsp页面
      i. 在课前资料中有login页面的全部资源，粘贴到webRoot目录下。
      ii. 修改login.html为login.jsp页面
      iii. 删除login.html
   b. 添加LoginServlet：

      代码如下：

```
import com.easymall.utils.JDBCUtils;
//登录功能
public class LoginServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //1.乱码处理
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        //2.获取用户信息
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        String remname = request.getParameter("remname");
        //如果remname值为true则要求记住用户名
        if("true".equals(remname)){
            //为true要求保存用户名
            Cookie cookie = new Cookie("remname",URLEncoder.encode(username,
            "utf-8"));
            cookie.setMaxAge(60*60*24*30);
            cookie.setPath(request.getContextPath()+"/");
            response.addCookie(cookie);
        }else{
            //如果不为true则应该销毁cookie
            Cookie cookie = new Cookie("remname","");
            cookie.setMaxAge(0);
            cookie.setPath(request.getContextPath()+"/");
            response.addCookie(cookie);
        }
        //3.访问数据库，比对用户信息
        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            conn = JDBCUtils.getConnection();
            ps = conn.prepareStatement("select * from user where username =? and password
            = ?");
```

```
                ps.setString(1, username);
                ps.setString(2, password);
                rs = ps.executeQuery();
                if(rs.next()){
                    //登录操作
                    //如果一致则保存用户登录信息，
                    //TODO:session
                    //4.回到首页
                    response.sendRedirect("http://www.easymall.com");
                }else{
                    //如果不一致则在页面中提示用户名或密码不正确
                    request.setAttribute("msg","用户名或者密码不正确");
                    request.getRequestDispatcher("/login.jsp").forward(request, response);
                    return;
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }finally{
                JDBCUtils.close(conn, ps, rs);
            }

        }

        public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            doGet(request, response);

        }

}
```

c. 修改login.jsp页面

i. 修改form标签

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

ii. 添加错误信息回显

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

iii. table中添加：

```
<%
    Cookie remnameC = null;
    Cookie[] cs = request.getCookies();
    if(cs != null){
        for(Cookie c:cs){
            if("remname".equals(c.getName())){
                remnameC = c;
            }
        }
    }
    String username = "";
    if(remnameC !=null){
```

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

    %>
```

iv. 在input[name='username']标签中添加

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

d. 回显选中记住用户名的对勾

在cookie中如果有用户的登录名称，则代表用户要求记住用户名，这时应该将记住用户名选框的对象勾选上。

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

会话技术--session

2019年5月21日 星期二 14:26

1. 在浏览器端存储数据的弊端：

   如果一个数据非常重要这个数据存储在浏览器端可以被任何一个用户查看到，无法保证当前数据的安全性，所以可以将数据由浏览器保存转向让服务器保存数据。

2. session实现原理

   a. 在浏览器访问服务器的时候，会在服务器内部创建一个与当前浏览器相关的session对象，在本次会话中如果用户多次请求服务器，都是操作的同一个session对象。不同浏览器会拥有各自的session对象。

   b. session的实现原理是通过一个JSESSIONID的cookie实现的。

3. session的使用

   a. request.getSession(); //如果服务器中已有浏览器对应的session，则这个方法会直接将session对象获取到拿来使用，如果服务器中没有对应的session，则这个方法会创建一个新的session对象使用。

   b. request.getSession(true);//如果服务器中已有浏览器对应的session，则这个方法会直接将session对象获取到拿来使用，如果服务器中没有对应的session，则这个方法会创建一个新的session对象使用。

   c. request.getSession(false);//如果服务器中没有浏览器对应的session，则会返回一个null的结果，如果有则拿过来使用。

4. session功能一：域对象

   域对象：在一个对象身上有一个可以被看见的范围，在这个范围内利用对象身上的map实现资源的共享，这样的对象就称之为域对象。

   a. 操作方法：

      ```
      setAttribute(String name,Object obj);
      getAttribute(String name);
      removeAttribute(String name);
      getAttributeNames();
      ```

   b. 生命周期

      request.getSession()方法调用的时候session对象创建。

      session对象的销毁:

      1) 超时死亡：session在默认情况下回会在服务器中存储30分钟，超过30会被服务器认为是一个无效session而销毁。超时时间可以在web.xml中配置一个<session-config>标签来指定。

      2) 自杀：当调用session.invalidate();会主动将session对象释放。

      3) 意外身亡：在非常关闭状态下，session会直接销毁。如果服务器正常关闭，则session对象会序列化到磁盘上成为一个文件，这个过程称之为钝化。在服务器下次启动的时候再将这个文件，读取到服务器当中，这个过程称之为活化。

   c. 作用范围

      当前会话范围内。

   d. 主要功能

当前会话范围内实现资源共享。

5. session案例：购物车

a. 需求：创建一个购物车的Servlet，和一个结账的Servlet，使两者可以实现购物车购买商品结账的需求.

b. 实现内容：在本次会话范围内实现用户购买商品数据的共享。

代码实现：

```
//购物车servlet
//商品添加的操作
public class BuyServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        String prod = request.getParameter("prod");
        prod = new String(prod.getBytes("iso8859-1"),"utf-8");

        //向session域中添加数据---在当前会话范围内实现商品数据的共享。
        //创建session
        HttpSession session = request.getSession();
        session.setAttribute("prod", prod);
        //创建一个JSESSIONID的cookie，将session保存在本地，
        //使浏览器多次关闭打开都能够获取到同一个session
        Cookie cookie = new Cookie("JSESSIONID",session.getId());
        cookie.setMaxAge(60*60*24);
        cookie.setPath(request.getContextPath()+"/");
        response.addCookie(cookie);

        //向页面显示商品添加信息。
        response.getWriter().write("已经将商品【"+prod+"】加入购物车");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);

    }

}
```

```
//付款servlet
//将商品计算价格并发送到浏览器
public class PayServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        //获取session对象
        HttpSession session = request.getSession();
        //获取session域中的属性
        String prod = (String) session.getAttribute("prod");
        if(prod == null){

            response.getWriter().write("您还未选择任何商品");
        }else{
```

```
            //向浏览器提示付款信息；
            response.getWriter().write("已付款商品【"+prod+"】，价格￥10000000");
        }
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doGet(request, response);

    }

}
```

6. EasyMall页面功能：
   a. 登录功能的实现：
      i. 修改login.jsp页面：在 中添加如下代码

```
<!--如果有session且session中包含username域属性则证明用户已经登录 -->
    <!-- 可以从session域中取出用户名 -->
    <%
        if(request.getSession(false)!=null &&
        request.getSession().getAttribute("username") != null){
    %>
    <a href="#">欢迎 <%=request.getSession().getAttribute("username") %> 回来
    </a>  |  
    <a href="<%=request.getContextPath() %>/LogOutServlet">注销</a>
    <%

        }else{
        %>
    <a href="<%=request.getContextPath()%>/login.jsp">登录</a>  |  
    <a href="<%=request.getContextPath() %>/regist.jsp">注册</a>
        <%
        }
     %>
```

      ii. 修改LoginServlet

```
//3.访问数据库，比对用户信息
Connection conn = null;
PreparedStatement ps = null;
ResultSet rs = null;
try {
    conn = JDBCUtils.getConnection();
    ps = conn.prepareStatement("select * from user where username =? and password = ?");
    ps.setString(1, username);
    ps.setString(2, password);
    rs = ps.executeQuery();
    if(rs.next()){
        //登录操作
        //如果一致则保存用户登录信息，
        //将用户的登录信息存储到session当中
        HttpSession session = request.getSession();
        session.setAttribute("username", username);
        //4.回到首页
        response.sendRedirect("http://www.easymall.com");
    }else{
        //如果不一致则在页面中提示用户名或密码不正确
        request.setAttribute("msg","用户名或者密码不正确");
```

```
            request.getRequestDispatcher("/login.jsp").forward(request, response);
            return;
        }
    }

    }
```

b. 注销功能实现：

i. 修改head.jsp页面

```
<a href="<%=request.getContextPath() %>/LogOutServlet">注销</a>
```

创建注销Servlet --- LogOutServlet

```
//注销用户登录状态
public class LogOutServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //注销用户登录状态可以将session删除，删除session后其中的属性值也会销毁
        //实现用户注销的功能
        request.getSession(false).invalidate();
        response.sendRedirect("http://www.easymall.com");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doGet(request, response);

    }

}
```

c. 验证码的实现：

i. request中存储验证码仅在当前request中有效，在下一次的去请求中会失效。保存数据的范围太小。

ii. ServletContext存储验证码,会在整个web应用的范围内，如果多个用户同时存储多个不同的验证码在web应用中，那么只有最后一个验证码会保存下来，会导致其他用户看到的图片验证码是真实验证码是不匹配。保存数据的范围过大。

iii. session将验证码存储在当前会话的范围内，可以在多次请求中使用，符合验证码的使用范围，而且session在多个浏览器直接也互补影响。

1） 修改RegistServlet;在//TODO:session位置添加如下代码

```
//验证码校验
        String code = (String) request.getSession().getAttribute("code");
        if(!valistr.equalsIgnoreCase(code)){
            request.setAttribute("msg", "验证码错误");
            request.getRequestDispatcher("/regist.jsp").forward(request, response);
            return;
        }
```

2） 修改validateServlet:

```
response.setDateHeader("Expires", -1);
        response.setHeader("Cache-control", "no-cache");
        //调用工具类产生一个验证码的图片
        VerifyCode vc = new VerifyCode();
        vc.drawImage(response.getOutputStream());
        //获取验证码的纯文本内容
        String code = vc.getCode();
```

```
//将创建出来的验证码加入到session域中。
request.getSession().setAttribute("code", code);
System.out.println(code);
```

Cookie和Session的特点

2019年5月21日 星期二 16:42

1. cookie特点：

   cookie是浏览器端的技术，保存在浏览器的数据可能会被其他人读取，造成数据丢失，数据安全性较差。cookie可以存储长期保存的数据。cookie适合存储安全性要求较低，而存储时间较长的数据。

2. session特点：

   session服务器端的技术，数据保存在服务器端安全性较高，普通用户无法读取服务器上的数据。session不善于长期存储数据，如果存储数据时间过长，会导致服务器长时间存储一个无人使用的对象，造成内存的浪费。session适合存储安全性较高，而存储时间较短的数据。