01_会话技术介绍

- 会话技术分为cookie与session
  - Cookie它是浏览器端的会话技术
  - Session它是服务器端的会话技术
- 问题:什么是会话技术？使用会话技术可以解决什么问题?
  - 会话可以简单理解成，通过浏览器访问服务器的资源，点击超连接可以进行资源的跳转，直到关闭浏览器过程叫做会话
  - 使用会话技术可以解决的是在整个会话过程中产生的数据的保存问题
- 之前数据存储存在的问题
  - request域
  - ServletContext域

02_Cookie介绍

- A.什么是Cookie?
  - Cookie是网景公司发明。Cookie是由服务器端创建，发送给浏览器，浏览器可以将cookie的key/value保存到浏览器端，下一次在请求同一个网站的资源时，就可以将cookie发送给服务器。
  - 在javaee的api中有一个类javax.servlet.http.Cookie。
- B.Cookie的执行流程
  - 浏览器向服务器发送请求,服务器需要创建cookie,服务器会通过响应携带cookie,在产生响应时,会产生set-cookie响应头,从而将cookie信息传递给了浏览器。
  - 当浏览器再次向服务器发送请求时,会产生cookie请求头,将之前服务器的cookie信息再次发送发送给了服务器,然后服务器会根据cookie信息跟踪客户端的状态.
- C.问题1:如何创建Cookie对象?
  - 查看api
- D.问题2:如何将Cookie响应到浏览器?
  - 方式1:
    - response.addHeader(“set-cookie”,”one=aaa”);
  - 方式2:
    - response.addCookie(Cookie);
- E.问题3:服务端如何获取到浏览器传递过来的Cookie?
  - 可以通过request的方法 getCookies()得到所有的cookie
- F.注意
  - Cookie不能存储中文

03_Cookie的相关设置

- A.持久化设置
  - 就是Cookie的生命周期设置。Cookie默认是在关闭浏览器就会销毁。
  - 可以通过setMaxAge(int expiry)方法设置

- B.路径设置
  - 确定当访问服务器的资源时，是否要携带cookie.
  - 可以通过setPath(String uri)方法设置
  - 比如:
    - cookie1的path路径为:/项目名称/
    - cookie2的path路径为:/项目名称/jsps/
    - 当访问http://localhost/项目名称/index.jsp 这时只有cookie1会被携带到服务器,因为url中包含了cookie1的path
    - 当我们访问http://localhost/项目名称/jsps/index.jsp 这时有cookie1和cookie2会被携带到服务器,因为url中同时包含了cookie1和cooki2的path

04_Cookie案例(显示上一次访问时间)

- A.需求
  - 当访问资源(ShowTimeServlet),第一次访问显示的第一次访问，时间xxx 如果第二次访问，显示的是上一次访问时间是:xxxx 以后每一次显示的都是上一次访问的时间。
- B.步骤
  - 获取Cookie对象
  - 判断是否第一次
    - 如果是第一次，直接显示当前时间
    - 如果不是第一次，显示上一次时间
  - 将当前时间存入到cookie

```
.....
if(null == cookie ){
    String currentTimeStr = new Date().toLocaleString();
    //之前没有访问过
    System.out.println("第一次访问，时间为："+ currentTimeStr);
} else {
    //之前有访问过
    String lastTimeStr = cookie.getValue();
    Date date = new Date();
    date.setTime(Long.parseLong(lastTimeStr));
    System.out.println("上一次的访问时间为："+date.toLocaleString());
}
cookie = new Cookie("lastVistedTime",new Date().getTime()+"");
response.addCookie(cookie);
```

05_Cookie案例(显示商品浏览记录)

- 商品页面

```
<a href="demo01?id=0">西游记</a>
<a href="demo01?id=1">红楼梦</a>
<a href="demo01?id=2">水浒传</a>
<a href="demo01?id=3">三国志</a>
```

- 加入浏览器记录

```
@WebServlet(name = "HistoryServlet" , urlPatterns = "/history")
public class HistoryServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        System.out.println("HistoryServlet");
        String id = request.getParameter("id");
        Cookie existCookie = null;
        Cookie[] cookies = request.getCookies();
        if (cookies != null & cookies.length != 0) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("history")) {
                    existCookie = cookie;
                }
            }
        }
        if (null != existCookie) {
            //记录不是空的
            String historyStr = existCookie.getValue();
            if (!historyStr.contains(id)){
                //没有该id的记录
                historyStr += "-" + id;
                existCookie .setValue(historyStr);
            } else {
                //有该id的记录
            }
        } else {
            //记录是空的
            existCookie = new Cookie("history",id);
        }
        //将cookie响应给浏览器存储
        response.addCookie(existCookie);
        //转发到显示历史浏览页面
        request.getRequestDispatcher("/showHistory").forward(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        doPost(request, response);
    }
}
```

- 显示浏览记录

```
@WebServlet(name = "ShowHistoryServlet" ,urlPatterns = "/showHistory")
public class ShowHistoryServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        Cookie existCookie = null;
        Cookie[] cookies = request.getCookies();
        if (cookies != null & cookies.length != 0) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("history")) {
                    existCookie = cookie;
                }
```

```
            }
        }
        String historyStr = existCookie.getValue();
        String[] historys = historyStr.split("-");
        String[] bookNames = {"水浒传","西游记","红楼梦","三国志"};
        StringBuffer sb = new StringBuffer();
        for (String history : historys) {
            sb.append("<font color='red'>"+bookNames[Integer.parseInt(history)]+"
</font><br>");
        }
        System.out.println(sb);
        response.setContentType("text/html;charset=utf-8");
        response.getWriter().write(sb.toString());
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        doPost(request,response);
    }
}
```

06_Session介绍

- Session是服务器端的会话技术，session其实是sun公司定义的一个接口 javax.servlet.http.HttpSession.
- 问题1:怎样获取HttpSession对象?
  - HttpServletRequest类中的getSession方法
- 问题2:Session的执行流程?
  - 第一次请求，我们的请求头中没有一个叫jsessionid的cookie，那么当访问到Demo1Servlet时，它执行request.getSession()就会创建一个HttpSession对象，当响应产生时，就将session的id做为cookie的value,响应到浏览器端 Set-cookie:jsessionid=xxxxxx
  - 再一次访问Demo2Servlet时，http请求中就会有一个cookie:jsessionid=xxxx信息，那么在Demo2Servlet中通过request.getSession()时会根据jsessionid的值在服务器内存中查找对应的session对象，如果有，直接拿过来使用，否则新建。
- Session的生命周期?
  - Session的相关信息是存储到Cookie中的，所以Session与Cookie是息息相关的，而Cookie默认情况下，关闭浏览器就会销毁，所以存储在Cookie中的相关Session信息也会随着销毁，session信息销毁并不代表session销毁，只是将cookie中存储的session对应的id销毁而已。
  - 而session的销毁默认是在30分钟内不使用会自动销毁，在tomcat的web.xml中可以看到有设置。
  - 同时，开发人员也可以通过方法去操作session的生命周期
    - invalidate():立即销毁session
    - setMaxInactiveInterval(int interval):在指定的时间内,如果不操作,session就销毁
    - 也可以在当前项目的web.xml中配置session-config

```
<session-config>
    <session-timeout>10</session-timeout>
```

```
</session-config>
```

07_登录案例之显示用户信息

- 需求：登录成功后，将用户信息存储到session，并跳转到首页显示用户信息。
- 登录功能

```
@WebServlet(name = "LoginServlet",urlPatterns = "/login")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        if ("root".equals(username) && "root".equals(password)) {
            //登录成功，重定向到首页
            User existUser = new User();
            existUser.setUsername(username);
            existUser.setPassword(password);
            request.getSession().setAttribute("existUser",existUser);
            //跳转到首页
            response.sendRedirect("/myweb/showIndex");
        } else {
            //登录失败，转发登录页面
            request.getRequestDispatcher("/login.html").forward(request,response);
        }
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        doPost(request, response);
    }
}
```

- 首页显示用户信息

```
@WebServlet(name = "ShowIndexServlet",urlPatterns = "/showIndex")
public class ShowIndexServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        HttpSession session = request.getSession();
        User existUser = (User) session.getAttribute("existUser");
        response.setContentType("text/html;charset=utf-8");
        if (null != existUser) {
            response.getWriter().write("欢迎回来!"+existUser.getUsername());
        } else {
            response.getWriter().write("您还没有登录，<a href='/myweb/login.html'>请登
录</a>");
        }

    }

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

```
    }
}
```

08_登录案例之加入验证码

- 登录页面
  - 静态验证码

```
<form action="/web01/login" method="post">
    ……
    验证码:<input type="text" id="code" onchange="checkCode()"/>
    <img   src="/web01/validatorCode.jsp">
    <span id="span1"></span>
    <br>
    ……
    <button type="submit">登录</button>
</form>
```

- 产生验证码(validatorCode.jsp)

```
int width = 60;
int height = 32;
//创建图片对象
BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();
//设置背景颜色
g.setColor(new Color(0xDCDCDC));
g.fillRect(0, 0, width, height);
//设置边框
g.setColor(Color.black);
g.drawRect(0, 0, width - 1, height - 1);
// create a random instance to generate the codes
Random rdm = new Random();
String hash1 = Integer.toHexString(rdm.nextInt());
//画干扰线
for (int i = 0; i < 50; i++) {
    int x = rdm.nextInt(width);
    int y = rdm.nextInt(height);
    g.drawOval(x, y, 0, 0);
}
//生成四位随机验证码
String capstr = hash1.substring(0, 4);
session.setAttribute("key", capstr);
System.out.println(capstr);
g.setColor(new Color(0, 100, 0));
g.setFont(new Font("Candara", Font.BOLD, 24));
g.drawString(capstr, 8, 24);
g.dispose();
response.setContentType("image/jpeg");
out.clear();
out = pageContext.pushBody();
```

```
OutputStream strm = response.getOutputStream();
ImageIO.write(image, "jpeg", strm);
strm.close();
```