# url编码和解码

|  |
| --- |
| String str = "你好";  str = URLEncoder.encode(str,"utf-8");  str = URLDecoder.decode(str,"utf-8"); |

# 2.Cookie

## 2.1引入

如果客户需要先登陆在去访问用户的主页那么有两种方式:

1)context域

登陆页面时将某个的数据保存到context域中

访问用户主页时,取出context域中的数据显示出来

但是如果第二个也这样的话,就出现问了,

使用context域会覆盖之前的数据

2)request域

登陆时将数据保存到request域中,此时如果要访问用户主页,

那么需要再次发送新的请求,此时这个请求并不包含登陆时保存的数据

所以不能重新发送请求,而是直接登陆后,通过转发技术跳转到用户主页上

这就有问题了,必须转发

以上两种都不适用

## 2.2会话管理技术

Cookie技术:会话数据保存在浏览器中

Session技术:会话数据保存在服务器中

### 2.2.1Cookie技术

Cookie对象:

1. 创建Cookie对象,用于储存对象

Cookie(String name,String value)

1. 修改Cookie对象

void setPath(String url)

void setMaxAge(int expiry)

void setValue(String newValue)

1. 把cookie数据发送给浏览器保存

response.addCookie(cookie);

1. 浏览器带着cokie访问服务器,服务器接受cookie信息

response.getCookies();

范例:

|  |
| --- |
| public class CookieDemo1 extends HttpServlet {  //使用cookie技术  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //1.创建cookie对象  Cookie cookie = new Cookie("name","jacky");  //2.把cookie发送给浏览器,通过响应头携带cookie数据给浏览器(set-cookie)  //response.setHeader("set-cookie","name=lw");  //简化版本  response.addCookie(cookie);  //3.浏览器在下次访问时候携带了Cookie数据,通过请求头发送给服务器(cookie字段)  //4.服务器获取浏览器发送的cookie数据  // String name = request.getHeader("cookie");  //简化版本  Cookie[] cookies = request.getCookies();  if(cookies != null){  for(Cookie ck: cookies){  String name = ck.getName();  String value = ck.getValue();  System.out.println(name + "=" + value);  }  }  }  } |

### 2.2.2Cookie技术原理

1) 服务器创建Cookie对象,保存会话数据,把Cookie数据发送给浏览器

response.addCookie(cookie);(响应头set-cookie:name=jack)

2) 浏览器获取cookie数据,保存在浏览器缓存区,然后在下次访问服务器时

携带Cookie数据（请求头:cookie:name=jack）

3)服务器获取浏览器发送的cookie数据 request.getCookie()

### 2.2.3Cookie细节

1) cookie的数据类型一定是字符串,发送中文的话要加密(但是不加密貌似也可以)

2） setPath(path):修改cookie所在的有效路径,有效路径:如果把cookie设置到某个路径下

，然后浏览器访问这个有效路径的时候,才会携带cookie数据给服务器,这个路径指的是类似/web路径,默认不设置路径,那么就会设置当前网站路径下web-apps下的web目录即/web

3 )setMaxAge(整数秒):设置cookie的有效时间

正整数:表示超过了正整数的数值时间,cookie就会丢失(cookie保存在浏览器的缓冲目录区中)

负整数:表示如果浏览器关闭了(而不是网页关了),cookie就会丢失(cookie保存在浏览器内存中)

0: 表示删除同名的cookie(0表示立即过期嘛,即删除同名cookie)

4) cookie可以有多个,但是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大小限制为4KB

### 2.3.4Cookie优点和缺点

缺点:

1)能保存的数据类型只能为字符串

2)保存在客户端,不安全,cookie大小有限制

优点:

1. 数据保存在客户端,减轻服务器的压力

# 3.Session

HttpSession对象:

1)创建HttpSession对象,用于保存会话数据

request.getSession();用户创建或者获取一个session对象

2)修改HttpSession对象:

void setMaxInactiveInterval(int interval);设置session对象的有效时间

void invalidate();手动销毁session对象

3)保存会话数据(作为域对象)

session.setAttribute(“name”,Object);

session.getAttribute(“name”);获取数据

session.removeAttribute(“name”)删除数据

|  |
| --- |
| public class SessionDemo1 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //1.创建HttpSession对象  HttpSession session = request.getSession();  //2.保存会话数据  session.setAttribute("name","jack");  }  } |

|  |
| --- |
| public class SessionDemo2 extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  //1.得到session对象  HttpSession session = request.getSession();  //2.获取会话数据  String name = (String)session.getAttribute("name");  if(name == null){  System.out.println("name is null");  }else{  System.out.println(name);  }  }  } |

## 3.1Session原理

1)服务器创建session对象,服务器会给这个session对象分配一个唯一的标记JSESSIONID

2)把JSESSIONIDID作为Cookie发送给浏览器

3)浏览器得到JSESSIONID保存下来,在下次访问携带这个JSESSIONIDID去访问服务器

4)服务器得到JSESSIONID,在服务器内存中搜索是否存在指定的JSESSIONID的session对象

5)如果找到,则返回这个session对象

6)如果找不到,可能直接返回null,或者再创建一个新的session对象.

## 3.2Session细节

1)setMaxInactiveInterval(秒数):设置session对象的有效时间

问题:session在什么时候销毁?

注意:不是浏览器关闭,session对象就销毁了!!!

默认情况下:等待30分钟空闲时间,session对象才会销毁,而setMaxInactiveInterval

就能改变这个时间，这个函数设置的是某个servlet的过期时间

，如果要设置全局的过期时间呢？

<web-app>

<!—全局session过期时间-->

<session-config>

<session-timeout>1</session-timeout> #单位为分钟

</session-config>

</web-app>

2)可以让JSESSIONID不会随着浏览器的关闭而丢失

|  |
| --- |
| //1.创建HttpSession对象  HttpSession session = request.getSession();  //设置JSESSION的时间,不会随着浏览器的关闭而消失  Cookie c = new Cookie("JSESSIONID",session.getId());  c.setMaxAge(1\*30\*24\*60\*60);  response.addCookie(c); |

3)直接手动销毁session对象

invalidate()

4)request.getSession();这里参数可以为false和ture,默认是true

如果是true:查询是否存在session对象,如果有,则返回,没有则创建session后返回

如果是false:查询是否存在session对象,如果有,则返回,如果没有则返回null