## Cookie与Session

Http是无状态的：同一个浏览器连接同一个网站，发送两次http请求，这两次请求都是独立的，没有任何关联。

造成的问题：无法记住密码，实现自动登录。即使登录之后，访问新的页面，需要从新填写密码

Cookie就是使用浏览器端的技术，实现在Cookie回话期间（同一个浏览器使用同一个Cookie），多次的响应/请求之间数据的共享

Session就是使用服务器技术，实现在Session会话期间（同一个服务器），多次请求/响应之间的数据共享

## Cookie的生命周期

Cookie.setMaxAge() 浏览器接受到set-Cookie响应头之后，会把浏览器端的cookie删除

如果设置为负数-1，则是会话cookie，正数则是持久化cookie，保存在本地磁盘中

会话Cookie，是把cookie存在ORM内存中的，关闭浏览器就失效。应用在session会话中

持久Cookie，是吧Cookie保存在本地磁盘中。实现免密码登录。

## 设置cookie对象的属性

Cookie cookie = new Cookie(String name, String value) 创建cookie对象

**Cookie的value中不能包含中文，否则会出现500错误。但是session中没有这个问题。**

设置有效路径：cookie.setPath(Stirng path).只有在访问这个路径，才会带有cookie请求头

设置有效域名：setDomain(Sting domain)只有在访问这个域名下的servlet，才会带有cookie的请求头。

设置http/https: setSecrue() true在https时候才带cookie头，false在http才带有cookie头

设置httpOnly: setHttpOnly(true), javaScript脚本无法读取cookie信息，防止XSS攻击。

将cookie对象添加到response中，http响应数据中才有set-cookie，浏览器端才会创建cookie

Response.addCookie(cookie) Response.addCookie(otherCookie)

读取cookie

Request.getCookies() 返回一个cookie数组。

## 创建cookie流程

入口程序Servlet1，创建cookie对象。

响应消息response中添加cookie对象

http响应数据的响应头中包含set-cookie

响应头到达浏览器，产生cookie

浏览器再次发送http请求的时候，请求头中带有符合条件的cookie。

Servlet程序读取请求头中的cookie，实现多次响应/请求之间的数据共享

## 重定向和请求转发中Cookie的区别

请求转发，访问Servlet-1，但是把请求转发到servlet\_2。Servlet\_2返回响应头和响应体。此时，由于Servlet\_1对象发response缓存，没有及时把包含set-cookie响应头发给浏览器，导致cookie的暂时失效

重定向：先返回响应，浏览器创建cookie，访问Servlet\_2

相同的cookie，但是设置了不同的有效路径，也会被当做是不同的cookie。

## Session的开启和删除

开启：Session request.getSession() 要么获得已有的Session，不存在就获得新的session。

删除：服务器自动删除一段时间内没有使用的session，或者手动删除sessio.invalidate()

浏览器自动退出，相应的session就会被删除

session.setAttribute(String name String value)

session.getAttribute(String name)

session.removeAttribute(String name)

## Session的创建

浏览器发送http请求，服务器创建session，session就此诞生

1. 创建唯一一个JsessionID，
2. 创建和JssionID对应的session对象。以便统一会话，不同的请求之间共享数据
3. 创建set-cookie JsessionID=66666

## Session流程

浏览器发送http请求：访问Servlet\_1，产生JsessionID=666，产生对应的session对象，创建set-cookie JssionId=666, 返回响应头。

第二次http请求，request.getSession() ，请求中的cookie中也有jssionId, 寻找对应的session，实现数据共享。