1. 会话技术
2. 概述

一次会话包含着多次请求和响应。当浏览器第一次给服务器发送请求时，会话建立，直到有一方断开为止。

1. 会话作用

在一次会话的范围内的多次请求间共享数据。

1. 方式
2. 客户端会话技术

Cookie。客户端会话技术就是将数据存储到客户端的会话技术。

1. 服务器端会话技术

Session。服务器端会话技术就是将数据存储到服务器端的会话技术。

1. Cookie
2. 概述

客户端的会话技术，将数据保存在客户端。

1. 使用步骤
2. 创建Cookie对象，绑定数据

new Cookie(String name, String value);

1. 发送Cookie对象

通过response响应对象来发送Cookie给浏览器：

response.addCookie(Cookie cookie);

1. 获取Cookie，获取数据

浏览器收到response对象发送的Cookie后，会保存在浏览器中。下一次发送请求时，会把Cookie带着一起发送，那么想要拿到请求中的Cookie，就要通过request来获取（可能会有多个Cookie，因此返回的是Cookie数组）：

Cookie[] request.getCookies()

1. Cookie原理分析

假设服务器端设置一个CookieDemo1发送Cookie，则在服务器端发送给浏览器端的response对象上设置一个响应头set-cookie，该响应头的值为Cookie的键值对，如set-cookie:msg=hello。

浏览器收到response对象后，将cookie的键值对保存，在下一次发送请求到服务器端时，同样会在request请求对象上设置一个请求头Cookie:msg=hello，这样就将浏览器中保存的cookie发送到服务器端了。

1. 发送多个Cookie

创建多个Cookie对象，并且为response对象调用多次addCookie方法，将多个Cookie发送出去。这样response对象中的响应头就会有多个set-cookie；而浏览器发送的request对象中的请求头中的Cookie的值就会有多个，用分号隔开。

如：Cookie:msg=hello;name=hehe

1. Cookie的生存周期
2. 默认情况

在默认情况下，Cookie存在于浏览器的内存中，当浏览器关闭后，Cookie数据被销毁。

1. 设置周期

为Cookie对象设置生存周期的方法为

setMaxAge(int seconds)

seconds为正数时：将Cookie数据写到硬盘中，持久化存储，seconds代表着Cookie的存活时间

seconds为负数时：设置Cookie的生存周期为默认值，即浏览器关闭Cookie被删除

seconds为0时：删除Cookie信息

1. Cookie的共享问题
2. 同一个Tomcat服务器中

同一个服务器中运行着多个项目，那么在不同项目中的Cookie是否能够共享呢？答案是可以的，不过默认情况下，每个Cookie都会被设置为当前项目的虚拟路径，即默认设置：

Cookie.setPath(项目的虚拟路径);

要使该Cookie能够在服务器的所有项目中共享，则手动设置路径为服务器的根路径即可：

Cookie.setPath(“/”);

1. 不同的Tomcat服务器中

在不同的Tomcat服务器中的Cookie也是可以共享的，只要设置一级域名相同即可。

如：

tieba.baidu.com和news.baidu.com的Cookie是可以共享的，只要设置相同的一级域名即可：

setDomain(“.baidu.com”)

1. Cookie的作用与特点
2. 作用
3. Cookie一般用于存储少量不太敏感的数据
4. 在不登录的情况下，完成服务器对客户端的身份识别
5. 特点
6. Cookie存储数据在客户端浏览器中
7. 浏览器对于单个Cookie的大小有限制（4Kb），对同一个域名下的总Cookie数量也有限制（20个）
8. Cookie中文问题

在Tomcat8之前，Cookie中不能直接存储中文数据，需要将中文数据转码，一般采用URL编码后，再通过URL解码显示中文数据。

在Tomcat8之后，Cookie可以支持中文数据，但是特殊字符依然不支持，需要使用URL编码并解码显示。

URL编码：

URLEncoder.encode(String s, String charset);

如：

URLEncoder.encode(str, “utf-8”);

1. Cookie实例
2. 需求
3. 访问一个Servlet，如果是第一次访问，则提示：您好，欢迎您首次访问
4. 如果不是第一次访问，则提示：欢迎回来，您上次访问时间为：xxx
5. 分析
6. 在服务器中的Servlet中判断是否有一个名为lastTime的Cookie
7. 如果有，则说明不是第一次访问：

响应数据：欢迎回来…；将新的时间Cookie写回到浏览器中

1. 如果没有，则说明是第一次访问：

响应数据：您好…；将新的时间Cookie写回到浏览器中

1. 注意事项

Cookie中的中文以及特殊字符问题需要使用URL编码以及解码来解决，如果报错500，原因是[32]，这说明是空格这一特殊字符的问题，将Cookie的值用URL编码后传输，在显示时用URL解码后显示。

1. 代码实现

package cookie;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.Cookie;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
import java.net.URLDecoder;  
import java.net.URLEncoder;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
  
@WebServlet("/cookieTest")  
public class CookieTest extends HttpServlet {  
  
 @Override  
 protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 //设置响应的消息体的数据格式以及编码  
 resp.setContentType("text/html;charset=utf-8");  
 //falg为false代表没有  
 boolean flag = false;  
 //获取所有的cookie  
 Cookie[] cookies = req.getCookies();  
 //遍历所有的cookie  
 if(cookies != null && cookies.length > 0){  
 for(Cookie cookie : cookies){  
 //获取Cookie的名称  
 String name = cookie.getName();  
 //判断lastTime是否存在  
 if("lastTime".equals(name)){  
 //存在  
 flag = true;  
 //设置Cookie的value  
 Date date = new Date();  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss");  
 String str\_date = sdf.format(date);  
 //打印编码前的字符串  
 System.*out*.println("编码前：" + str\_date);  
 //URL 编码  
 str\_date = URLEncoder.*encode*(str\_date, "utf-8");  
 System.*out*.println("编码后："+ str\_date);  
  
 cookie.setValue(str\_date);  
 //设置Cookie的存活时间  
 cookie.setMaxAge(60 \* 60 \* 24 \* 30);  
 resp.addCookie(cookie);  
  
  
 //获取Cookie的value  
 String value = cookie.getValue();  
 //URL解码  
 System.*out*.println("解码前："+value);  
 value = URLDecoder.*decode*(value, "utf-8");  
 System.*out*.println("解码后："+value);  
 //写出数据到resp上  
 resp.getWriter().write("<h1>欢迎回来，您上次访问时间为："+ value + "</h1>");  
 //找到直接break  
 break;  
 }  
 }  
 }  
 if(cookies == null || cookies.length==0 || flag == false){  
 //第一次访问  
 //设置Cookie的value为当前的时间  
 Date date = new Date();  
 SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm:ss");  
 String str\_data = sdf.format(date);  
 System.*out*.println("编码前" + str\_data);  
 //URL编码  
 str\_data = URLEncoder.*encode*(str\_data, "utf-8");  
 System.*out*.println("编码后：" + str\_data);  
  
 //new一个cookie为lastTime登录时间  
 Cookie cookie = new Cookie("lastTime", str\_data);  
 //设置Cookie的存活时间  
 cookie.setMaxAge(60 \* 60 \* 24 \* 30);  
 resp.addCookie(cookie);  
 resp.getWriter().write("<h1>您好，欢迎您首次访问</h1>");  
 }  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {  
 this.doPost(req, resp);  
 }  
}