* 什么是会话?

当用户开一个浏览器访问某个网站,点击多个超链接,访问服务器的多个资源直到关闭浏览器,整个过程我们称为一次会话.

* 会话中要解决的问题

会话会产生数据,服务器要要想办法存储这些数据

分类:

Cookie技术

Session技术

为什么需要cookie技术(会话技术)

如何保存用户上次登录时间



如何显示用户浏览历史



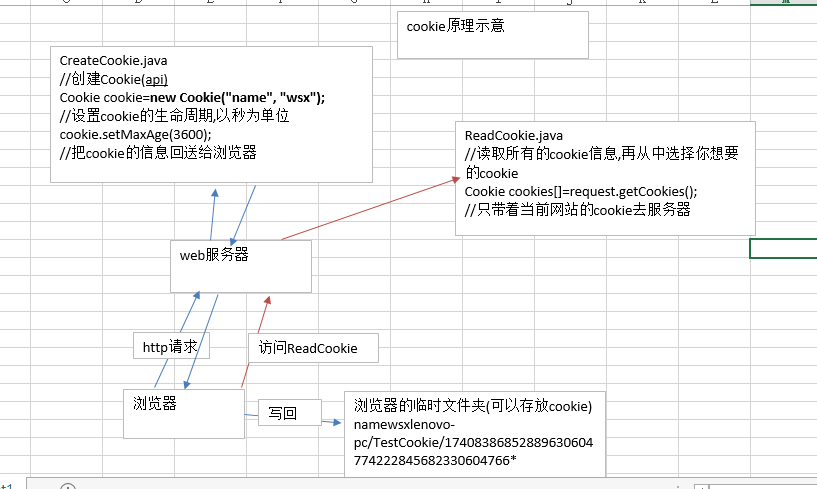
如何把登陆的用户名密码保存到电脑上,下次登录无需填写



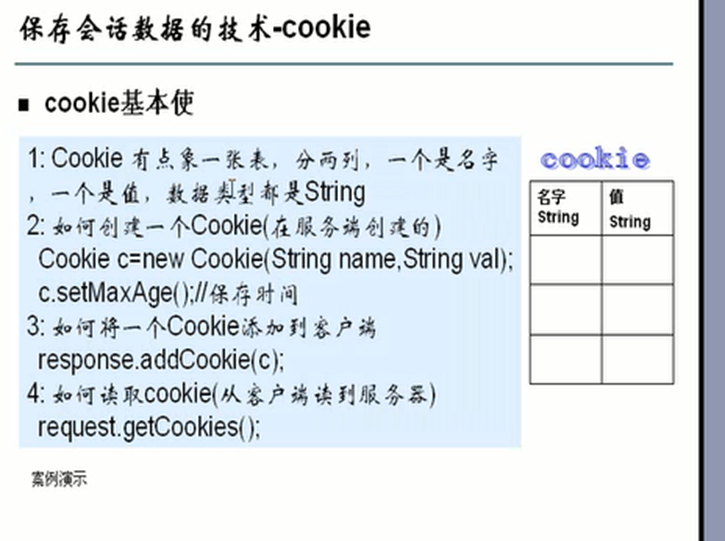
Cookie能解决

客户端技术,服务器把每个用户的数据以cookie的形式写给用户各自的浏览器,当用户用浏览器再去访问服务器中的web资源师就会带着各自的数据去

* Cookie的原理图



基本使用



Cookie小结

1. cookie是在服务器端创建的
2. cookie是保存在浏览器这端的
3. cookie的生命周期可以通过

cookie.setMaxAge(2000)设置,单位为秒

如果不设置setMaxAge,则该cookie在浏览器关闭时就消亡

1. cookie可以被多个浏览器共享
2. 理解:我们可以把cookie理解为一张表(如上图)

如果重名就会替换原来的cookie值,写入最新的

1. 一个web应用可以保存多个cookie
2. Cookie存放的时候是以明文方式存放的,因此安全性较低,我们可以通过加密后再保存.

补讲一个md5算法:以后密码都用md5加密存放,再验证密码时候,对用户输入密码进行md5加密与数据库中的比对

Cookie的实际使用

* 显示上次登录时间
* 自动填充用户名密码

Cookie的细节讨论

1. 一个浏览器最多放300个cookie,一个web站点最多放20个cookie,而且cookie的大小限制在4kb之内
2. Cookie的生命周期再说明
3. Cookie的默认生命周期是会话级别的,退出浏览器就删除
4. 通过setMaxAge()可以设置生命周期

setMaxAge(正数) ,即多少秒后cookie失效

setMaxAge(0),删除该cookie

案例:

Cookie cookies[]=request.getCookies();

**for**(Cookie cookie:cookies){

**if**(cookie.getName().equals("password")){

//System.out.println("delete");

cookie.setMaxAge(0);//删除

response.addCookie(cookie);//一定要加,不然删除不掉

特别说明:如果web应用只有一个cookie,则删除该cookie后,在浏览器的临时文件夹中的cookie文件都被删除,,如果该web应用有多个cookie,则删除一个cookie后,,只删除那个cookie,其他的还在

setMaxAge(负数),相当于默认生命周期,浏览器关闭就退出

* Cookie中写入中文

//向cookie中写入中文,需要对中文进行编码,再写入

String val=java.net.URLEncoder.*encode*("魏少轩", "utf-8");

//得到值后解码

String val=java.net.URLDecoder.*decode*(cookie.getValue(), "utf-8");

* out.print("name的属性"+val);Session为什么有?

问题一:如何实现在不同页面,可以去查看信息

* Session保存在服务器端

当用户打开浏览器访问某网站时,服务器就会在服务器内存中为用户分配一个session对象

Session小结

1. session是存放在服务器的内存中的
2. session中属性的默认生命周期是30分钟,你可以通过Tomcat的web.xml文件来修改

修改方法:

1. 直接在Tomcat中修改,所有的web应用都修改

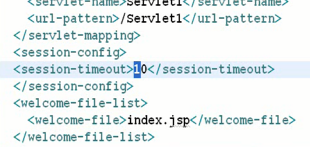
<session-config>

<session-timeout>30</session-timeout>

</session-config>

1. 在单个的web应用下去修改

在项目内的web.xml修改



如果发生冲突,就以自己的web.xml为准

c. session.setMaxInactiveInterval(60)

session.setMaxInactiveInterval(60);

//这个精确到秒的时间.是发呆时间,也就是若60秒不调用这个session,该session就被清除,

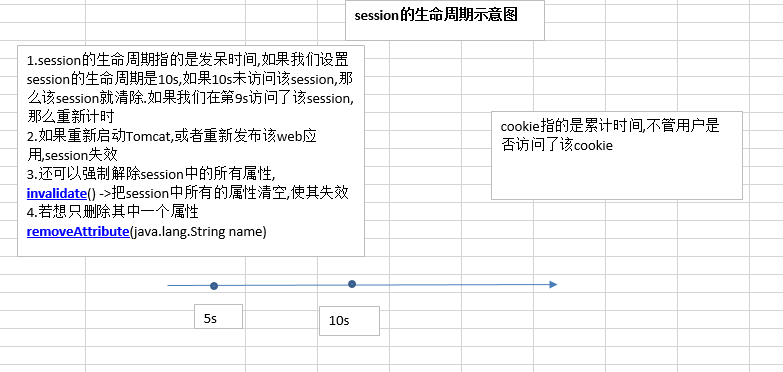
//如果60内调用了,那么重新计时

1. session中可以存放多个属性
2. session可以存放对象
3. 如果session的名字重复了,则会替换该名字的值
4. 一个浏览器独享一个session对象
5. 因为session一旦被访问,如果没有就会被创建

getSession(boolean create) 如果参数值是false,访问若没访问到就不会创建session

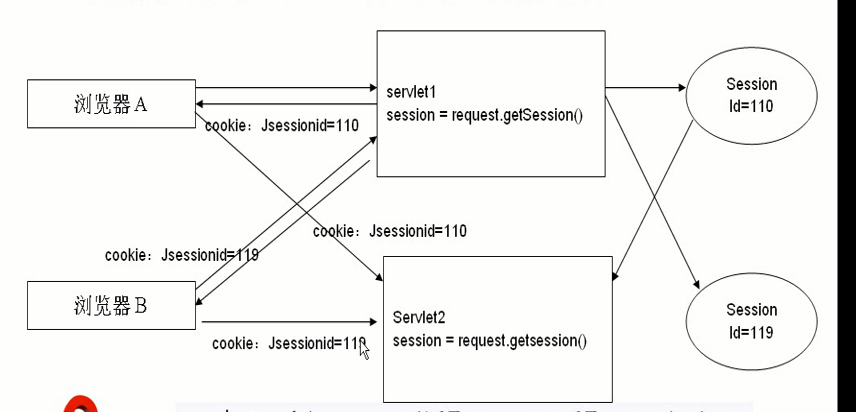
1. 例如: getSession(false)

* Session生命周期小结



?如果同一个用户浏览器,向session设置一个属性的时候,名字相同了会发生什么

会替换原来的session值



第一次请求session,浏览器的请求没有带任何东西,服务器就创建了一个session,同时把session的ID号(JSESSIONID)以cookie的方式保存在了浏览器的临时文件中.

第二次请求session ,浏览器的请求中就带上了上次创建session时服务器回送的session的的ID号(JSESSIONID),凭借这个JSESSIONID服务器才能找到之前该浏览器在自己的内存中创建的session

[www.sourceforge.net](http://www.sourceforge.net) [开源之祖]

* 验证码案例

原理是Java的绘图技术

生成随机验证码图片的CreateCode类

package com.wsx;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.util.Random;

import javax.imageio.ImageIO;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class CreateCode extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

//7.禁止浏览器缓存随机图片

response.setDateHeader("Expires", -1);

response.setHeader("Cache-Control", "no-cache");

response.setHeader("Pragma", "no-cache");

//6.通知客户机以图片的方式打开发送过去的数据

response.setHeader("Content-Type", "image/jpeg");

//1.在内存中创建一个图片

BufferedImage image=new BufferedImage(80, 30,

BufferedImage.TYPE\_INT\_RGB);

//2.向图片上写数据

Graphics g= image.getGraphics();

//设置背景色

g.setColor(Color.BLACK);

g.fillRect(0, 0, 80, 30);

//3.设置写入数据的颜色和字体

g.setColor(Color.GREEN);

g.setFont(new Font(null, Font.BOLD, 20));

//4.向图片上写数据

String num=makeNum();

//生成随机数字并写道session中去

request.getSession().setAttribute("checkcode", num);

g.drawString(num, 0, 20);

//5.把写好数据的图片输出给浏览器

ImageIO.write(image, "jpg", response.getOutputStream());

}

//该函数随机生成7位数字

public String makeNum() {

// TODO Auto-generated method stub

Random r=new Random();

//9999999 最大可以生成7位数

String num=r.nextInt(9999999)+"";

StringBuffer sb=new StringBuffer();

//如果生成的数字不够7位,那么在前面补零

for(int i=0;i<7-num.length();i++){

sb.append("0");

}

num=sb.toString()+num;

return num;

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

doGet(request, response);

}

}

如何使用产生的随机验证码图片

Login.java

<img src=’/验证码的URL’>

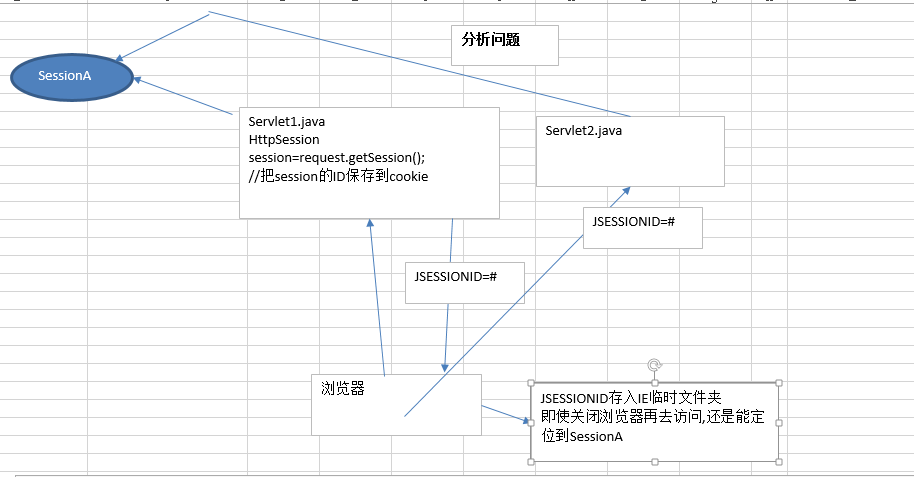
对session的销毁时间的讨论->借助案例

面试题;(应用:关闭掉IE后,再开IE,上次购买的商品还在->涉及到session的销毁时间)

分析:

* Session的生命周期若是30分钟,该session不会随着浏览器的关闭而消失,他会等到30分钟后再被销毁

分析实践的思路:



简易购物车

当用户点击商品时,我们把该商品保存到session中

session的结构是

Name value

Mybooks HashMap对象

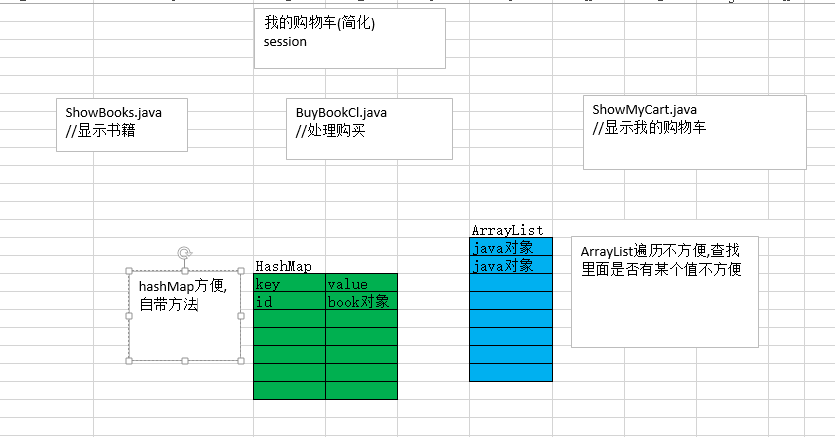
Hashmap的结构

Key value

书号 书对象

Book.java类

见MyCart项目



总结:

1. java基础的集合ArrayList() HashMap() LinkedHashMap()(有序的)
2. session技术
3. Servlet
4. 如何选择不同的集合

List集合都是有序的集合,Map集合默认是无序集合

List和Map集合都可以放入空值,list可以放相同的对象,map也可以放入相同对象,但是key值不能重复

* Cookie 和session大总结

1. 存在的位置

Cookie存在客户端浏览器的临时文件夹,多个浏览器可以共享

Session存在服务器的内存中,每个用户独享一个域对象

1. 安全性

Cookie是以明文方式存放在客户端的,安全性相对弱一点,可以加密再存放(MD5)

Session是存放在服务器内存中的,所以安全性相对较强

1. 网络传输量

Cookie会传递信息给服务器

Session不会给客户端传递属性值

1. 生命周期

Cookie的生命周期,是累计时间,即如果我们给cookie设置了setMaxAge(30),则30秒后失效

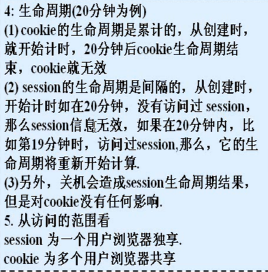
Session的生命周期,是间隔时间,例如我们设置session 20min,指的是在20分钟内如果没有访问session,则session失效(无法取出session的属性值),在以下情况session也会失效:

1.关闭Tomcat

2.reloadweb应用

3.时间到

4.invalidate() :解除所有绑定在该session上的对象,让session失效



* Session的使用原则:session会占用服务器的内存,因此不要向session中存放过多过大的对象,影响性能