**Web开发入门**

# Web开发入门

## 引入

之前的程序： java桌面程序，控制台控制，socket gui界面。javase规范

现在和以后的程序：java web程序。浏览器控制。javaee规范

## 软件的结构

C/S (Client - Server 客户端-服务器端)

典型应用：QQ软件 ，飞秋，红蜘蛛。

特点：

1）必须下载特定的客户端程序。

2）服务器端升级，客户端升级。

B/S （Broswer -Server 浏览器端- 服务器端）

典型应用： 腾讯官方（www.qq.com） 163新闻网站， 蚂蚁课堂官网（俗称：网站）

特点：

1）不需要安装特定的客户端（只需要安装浏览器即可！！）

2）服务器端升级，浏览器不需要升级！！！！

javaweb的程序就是b/s软件结构！！！

## 服务器

从物理上来说，服务器就是一台PC机器。8核，8G以上，T来计算，带宽100M

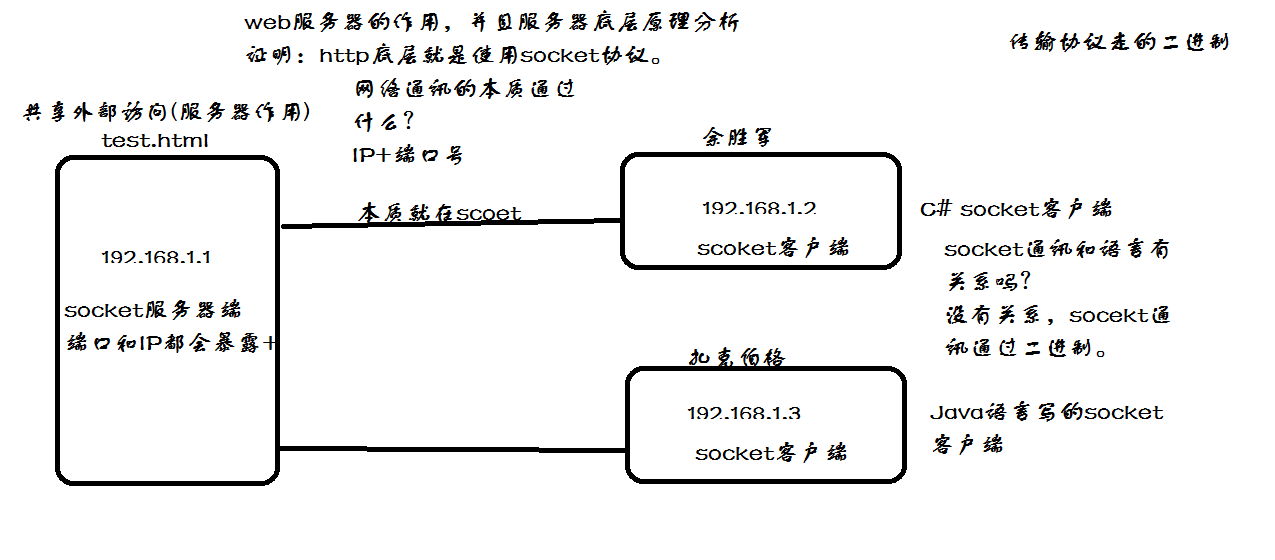
web服务器：PC机器安装一个具有web服务的软件，称之为web服务器

数据库服务器：PC机器安装一个具有数据管理件服务的软件，称之为数据库服务器。

邮件服务器：PC机器安装一个具有发送邮件服务的软件，称之为邮件服务器。

## web服务软件

web服务软件的作用：把本地的资源共享给外部访问。



## 常见的市面上web服务软件

WebLogic: BEA公司的产品。 收费的。支持JavaEE规范。

WebSphere： IBM公司的产品。收费的。支持JavaEE规范

JBoss: Redhat公司的产品。收费的。支持JavaEE规范

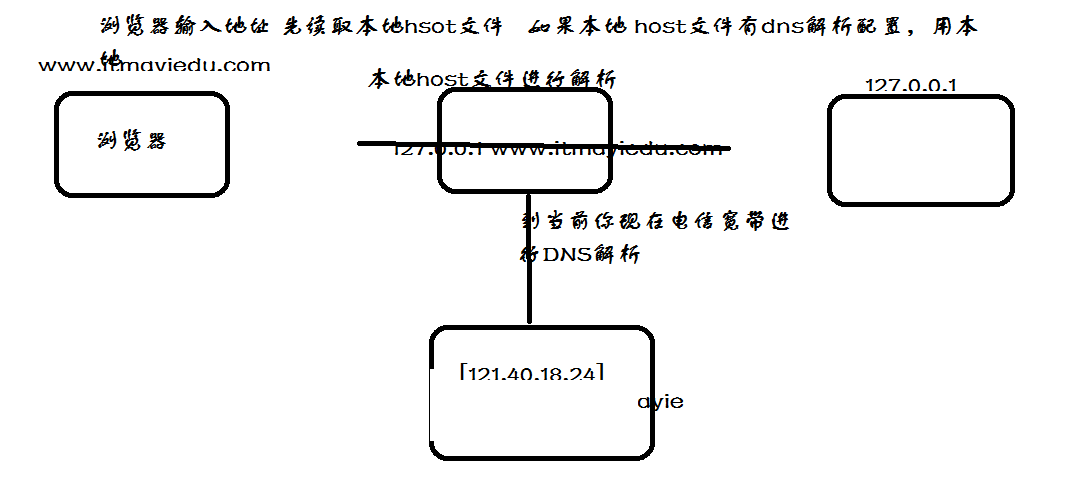
Tomcat： 开源组织Apache的产品。免费的。支持部分的JavaEE规范。（servlet、jsp。jdbc，但 ejb， rmi不支持）

# DNS解析域名过程

使用域名转换成IP地址，先读取本地HOST文件，本地文件没有从当前电信网管获取对应IP。

本地host文件 C:\Windows\System32\drivers\etc

画图演示。



# 内网与外网

## 内网与外网的区别？

[外网](https://www.baidu.com/s?wd=%E5%A4%96%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)、[内网](https://www.baidu.com/s?wd=%E5%86%85%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)是两种Internet的接入方式。  
　　[内网](https://www.baidu.com/s?wd=%E5%86%85%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)通俗的说就是[局域网LAN](https://www.baidu.com/s?wd=%E5%B1%80%E5%9F%9F%E7%BD%91LAN&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)网，[外网](https://www.baidu.com/s?wd=%E5%A4%96%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)通俗的说就是与因特网相通的WAN[广域网](https://www.baidu.com/s?wd=%E5%B9%BF%E5%9F%9F%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)或 MAN [城域网](https://www.baidu.com/s?wd=%E5%9F%8E%E5%9F%9F%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)路。[内网](https://www.baidu.com/s?wd=%E5%86%85%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)和[外网](https://www.baidu.com/s?wd=%E5%A4%96%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)是相对而言的。一般外网的范围比内网大，也可以说内网是外网的子网。  
　　外网（[广域网](https://www.baidu.com/s?wd=%E5%B9%BF%E5%9F%9F%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)）上的每一台电脑（或其他网络设备）都有一个或多个[广域网](https://www.baidu.com/s?wd=%E5%B9%BF%E5%9F%9F%E7%BD%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)[IP地址](https://www.baidu.com/s?wd=IP%E5%9C%B0%E5%9D%80&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)（或者说公网、外网[IP地址](https://www.baidu.com/s?wd=IP%E5%9C%B0%E5%9D%80&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)），广域网[IP地址](https://www.baidu.com/s?wd=IP%E5%9C%B0%E5%9D%80&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y1m1nsPvR4nHmkuWnLnyPW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnH6zrHbznHcvn1mdnjD4nHT3n0" \t "https://zhidao.baidu.com/question/_blank)不能重复；局域网（LAN）上的每一台电脑（或其他网络设备）都有一个或多个局域网IP地址（或者说私网、内网IP地址），局域网IP地址是局域网内部分配的，不同局域网的IP地址可以重复，不会相互影响。

## 外网映射工具

在做一些支付项目、微信开发、或对接第三方接口的时候，有些回调操作，可能会需要外网访问。

那么我们在做本地开发的时候，外网无法访问到本地？

怎么解决这个问题呢？

就会用到外网映射工具,常用外网映射工具netapp(免费)、花生壳等。

参考资料: https://natapp.cn/article/natapp\_newbie

# Servlet核心内容

## Sevlet的生命周期（重点）

### Servlet重要的四个生命周期方法

构造方法： 创建servlet对象的时候调用。默认情况下，第一次访问servlet的时候创建servlet对象只调用1次。证明servlet对象在tomcat是单实例的。

init方法： 创建完servlet对象的时候调用。只调用1次。

service方法： 每次发出请求时调用。调用n次。

destroy方法： 销毁servlet对象的时候调用。停止服务器或者重新部署web应用时销毁servlet对象。只调用1次。

### 伪代码演示servlet的生命周期

Tomtcat内部代码运行：

1）通过映射找到到servlet-class的内容，字符串： com.itmayiedu.a\_servlet.FirstServlet

2）通过反射构造FirstServlet对象

2.1 得到字节码对象

Class clazz = class.forName("com.itmayiedu.a\_servlet.FirstServlet");

2.2 调用无参数的构造方法来构造对象

Object obj = clazz.newInstance(); ---1.servlet的构造方法被调用

3）创建ServletConfig对象，通过反射调用init方法

3.1 得到方法对象

Method m = clazz.getDeclareMethod("init",ServletConfig.class);

3.2 调用方法

m.invoke(obj,config); --2.servlet的init方法被调用

4）创建request，response对象，通过反射调用service方法

4.1 得到方法对象

Methodm m =clazz.getDeclareMethod("service",HttpServletRequest.class,HttpServletResponse.class);

4.2 调用方法

m.invoke(obj,request,response); --3.servlet的service方法被调用

5）当tomcat服务器停止或web应用重新部署，通过反射调用destroy方法

5.1 得到方法对象

Method m = clazz.getDeclareMethod("destroy",null);

5.2 调用方法

m.invoke(obj,null); --4.servlet的destroy方法被调用

### 用时序图来演示servlet的生命周期



## Servlet的多线程并发问题

注意： servlet对象在tomcat服务器是单实例多线程的。

因为servlet是多线程的，所以当多个servlet的线程同时访问了servlet的共享数据，如成员变量，可能会引发线程安全问题。

解决办法：

1）把使用到共享数据的代码块进行同步（使用synchronized关键字进行同步）

2）建议在servlet类中尽量不要使用成员变量。如果确实要使用成员，必须同步。而且尽量缩小同步代码块的范围。（哪里使用到了成员变量，就同步哪里！！），以避免因为同步而导致并发效率降低。

Servlet学习：

HttpServletRequest 请求对象：获取请求信息

HttpServletResponse 响应对象： 设置响应对象

ServletConfig对象 servlet配置对象

ServletContext对象； servlet的上下文对象

### 线程安全代码:

|  |
| --- |
| **package** com.servlet;  **import** java.io.IOException;  **import** java.util.Date;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  /\*\*  \*  \* **@classDesc**: 功能描述:(线程安全演示)  \* **@author**: 余胜军  \* **@createTime**: 2017年8月31日 下午11:47:20  \* **@version**: v1.0  \* **@copyright**:上海每特教育科技有限公司  \*/  **public** **class** ServetlDemo4 **extends** HttpServlet {  **private** **int** i = 1;  @Override  **public** **void** init() **throws** ServletException {  System.***out***.println("ServetlDemo4...init()");  }  @Override  **protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {  // 设置编码格式  // resp.setContentType("text/html;charset=utf-8");  resp.setCharacterEncoding("utf-8");// 内容编码，防止出现中文乱码  resp.setContentType("text/html;charset=utf-8");  **synchronized** (ServetlDemo4.**class**) {  // 向浏览器输出内容  resp.getWriter().write("这是第" + i + "次访问...");  **try** {  Thread.*sleep*(5000);  } **catch** (Exception e) {  // **TODO**: handle exception  }  i++;  }  }  @Override  **public** **void** destroy() {  System.***out***.println("ServetlDemo4...destroy()");  }  } |

# 会话管理入门

## 生活中会话

我： 小张，你会跳小苹果码？

小张： 会，怎么了？

我： 公司年会上要表演节目，你教教我把

小张：没问题，一顿饭而已。

我： OK。

。。。。。。。。

在这次生活中的会话中产生通话记录（会话数据）

## 软件中的会话

一次会话： 打开浏览器 -> 访问一些服务器内容 -> 关闭浏览器

登录场景：

打开浏览器 -> 浏览到登陆页面 -> 输入用户名和密码 -> 访问到用户主页(显示用户名)

修改密码（输入原密码）

修改收货地址

.......

问题：在此处登录会话过程中产生的数据（用户会话数据）如何保存下来呢？

购物场景：

打开浏览器 -> 浏览商品列表 -> 加入购物车(把商品信息保存下来) -> 关闭浏览器

打开浏览器-> 直接进入购物车 -> 查看到上次加入购物车的商品 -> 下订单 -> 支付

问题： 在购物会话过程中，如何保存商品信息？？

会话管理： 管理浏览器客户端 和 服务器端之间会话过程中产生的会话数据。

域对象： 实现资源之间的数据共享。

request域对象

context域对象

登录场景：

小张： 输入“张三” （保存数据： context.setAttribute("name","张三")） -> 用户主页（显示“张三”）

小李： 输入“李四”(保存数据：context.setAttribute("name","李四")) -> 用户主页（显示“李四”）

问题： context是所有用户公有的资源！！！会覆盖数据。

小张： 输入“张三”（保存数据： request.setAttribute("name","张三")）- > 用户主页（显示“张三”） 问题： 一定要使用转发技术来跳转页面！！！

解决办法： 可以使用session域对象来保存会话数据！！！

## 会话技术

Cookie技术：会话数据保存在浏览器客户端。

Session技术：会话数据保存在服务器端。

# Cooke技术

## 特点

Cookie技术：会话数据保存在浏览器客户端。

## Cookie技术核心

Cookie类：用于存储会话数据

1）构造Cookie对象

Cookie(java.lang.String name, java.lang.String value)

2）设置cookie

void setPath(java.lang.String uri) ：设置cookie的有效访问路径

void setMaxAge(int expiry) ： 设置cookie的有效时间

void setValue(java.lang.String newValue) ：设置cookie的值

3）发送cookie到浏览器端保存

void response.addCookie(Cookie cookie) : 发送cookie

4）服务器接收cookie

Cookie[] request.getCookies() : 接收cookie

## Cookie原理

1）服务器创建cookie对象，把会话数据存储到cookie对象中。

new Cookie("name","value");

2） 服务器发送cookie信息到浏览器

response.addCookie(cookie);

举例： set-cookie: name=eric (隐藏发送了一个set-cookie名称的响应头)

3）浏览器得到服务器发送的cookie，然后保存在浏览器端。

4）浏览器在下次访问服务器时，会带着cookie信息

举例： cookie: name=eric (隐藏带着一个叫cookie名称的请求头)

5）服务器接收到浏览器带来的cookie信息

request.getCookies();

## Cookie的细节

1）void setPath(java.lang.String uri) ：设置cookie的有效访问路径。有效路径指的是cookie的有效路径保存在哪里，那么浏览器在有效路径下访问服务器时就会带着cookie信息，否则不带cookie信息。

2）void setMaxAge(int expiry) ： 设置cookie的有效时间。

正整数：表示cookie数据保存浏览器的缓存目录（硬盘中），数值表示保存的时间。

负整数：表示cookie数据保存浏览器的内存中。浏览器关闭cookie就丢失了！！

零：表示删除同名的cookie数据

3）Cookie数据类型只能保存非中文字符串类型的。可以保存多个cookie，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的大小限制为4KB。

# Session技术

## 引入

Cookie的局限：

1）Cookie只能存字符串类型。不能保存对象

2）只能存非中文。

3）1个Cookie的容量不超过4KB。

如果要保存非字符串，超过4kb内容，只能使用session技术！！！

Session特点：

会话数据保存在服务器端。（内存中）

## Session技术核心

HttpSession类：用于保存会话数据

1）创建或得到session对象

HttpSession getSession()

HttpSession getSession(boolean create)

2）设置session对象

void setMaxInactiveInterval(int interval) ： 设置session的有效时间

void invalidate() ： 销毁session对象

java.lang.String getId() ： 得到session编号

3）保存会话数据到session对象

void setAttribute(java.lang.String name, java.lang.Object value) ： 保存数据

java.lang.Object getAttribute(java.lang.String name) ： 获取数据

void removeAttribute(java.lang.String name) ： 清除数据

## Session原理

问题： 服务器能够识别不同的浏览者！！！

现象：

前提： 在哪个session域对象保存数据，就必须从哪个域对象取出！！！！

浏览器1：(给s1分配一个唯一的标记：s001,把s001发送给浏览器)

1）创建session对象，保存会话数据

HttpSession session = request.getSession(); --保存会话数据 s1

浏览器1 的新窗口（带着s001的标记到服务器查询，s001->s1,返回s1）

1）得到session对象的会话数据

HttpSession session = request.getSession(); --可以取出 s1

新的浏览器1：(没有带s001,不能返回s1)

1）得到session对象的会话数据

HttpSession session = request.getSession(); --不可以取出 s2

浏览器2：(没有带s001,不能返回s1)

1）得到session对象的会话数据

HttpSession session = request.getSession(); --不可以取出 s3

代码解读：HttpSession session = request.getSession();

1）第一次访问创建session对象，给session对象分配一个唯一的ID，叫JSESSIONID

new HttpSession();

2）把JSESSIONID作为Cookie的值发送给浏览器保存

Cookie cookie = new Cookie("JSESSIONID", sessionID);

response.addCookie(cookie);

3）第二次访问的时候，浏览器带着JSESSIONID的cookie访问服务器

4）服务器得到JSESSIONID，在服务器的内存中搜索是否存放对应编号的session对象。

if(找到){

return map.get(sessionID);

}

Map<String,HttpSession>]

<"s001", s1>

<"s001,"s2>

5）如果找到对应编号的session对象，直接返回该对象

6）如果找不到对应编号的session对象，创建新的session对象，继续走1的流程

结论：通过JSESSION的cookie值在服务器找session对象！！！！！

## Sesson细节

1）java.lang.String getId() ： 得到session编号

2）两个getSession方法：

getSession(true) / getSession() : 创建或得到session对象。没有匹配的session编号，自动创 建新的session对象。

getSession(false): 得到session对象。没有匹配的session编号，返回null

3）void setMaxInactiveInterval(int interval) ： 设置session的有效时间

session对象销毁时间：

3.1 默认情况30分服务器自动回收

3.2 修改session回收时间

3.3 全局修改session有效时间

|  |
| --- |
| <!-- 修改session全局有效时间:分钟 -->  <session-config>  <session-timeout>1</session-timeout>  </session-config> |

3.4.手动销毁session对象

void invalidate() ： 销毁session对象

4）如何避免浏览器的JSESSIONID的cookie随着浏览器关闭而丢失的问题

|  |
| --- |
| /\*\*  \* 手动发送一个硬盘保存的cookie给浏览器  \*/  Cookie c = **new** Cookie("JSESSIONID",session.getId());  c.setMaxAge(60\*60);  response.addCookie(c); |

总结：

1）会话管理： 浏览器和服务器会话过程中的产生的会话数据的管理。

2）Cookie技术：

new Cookie("name","value")

response.addCookie(coookie)

request.getCookies()

3）Session技术

request.getSession();

setAttrbute("name","会话数据");

getAttribute("会话数据")