会话跟踪

本章内容：共3小节，11个知识点

- 第1节：会话跟踪概述
- 第2节： Cookie
- 第3节： Session

本章目标

- 掌握会话跟踪机制的概念、作用；
- 能够使用Cookie实现简单登录等功能；
- 熟悉Servlet API中会话有关的接口；
- 熟练掌握会话中的常用功能；

第1节【会话跟踪概述】

- 知识点1：会话的概念与作用
- 知识点2：现行常用的会话跟踪技术

知识点1-会话的概念与作用-1

- 对于Web应用的一个关注核心：

WEB应用最关心的：

你是谁-你要干什么-你要到哪里去?

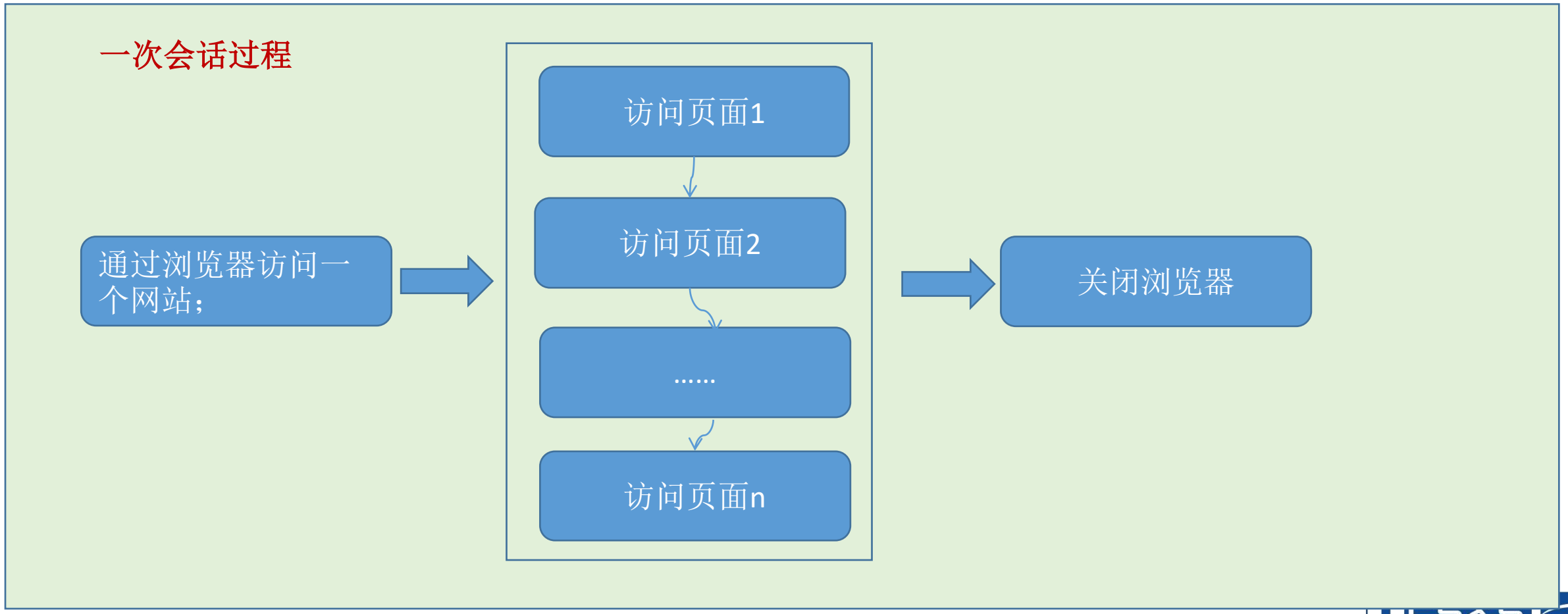
好心塞，如何告诉服务器，我是谁，我要干什么，我要到哪里去

知识点1-会话的概念与作用-1

- 在无状态的HTTP协议下要想直接完成这个任务...

知识点1-会话的概念与作用-1

- 对于Web应用来说，会话（Session）就是浏览器与服务器之间的一次连续的通讯过程；



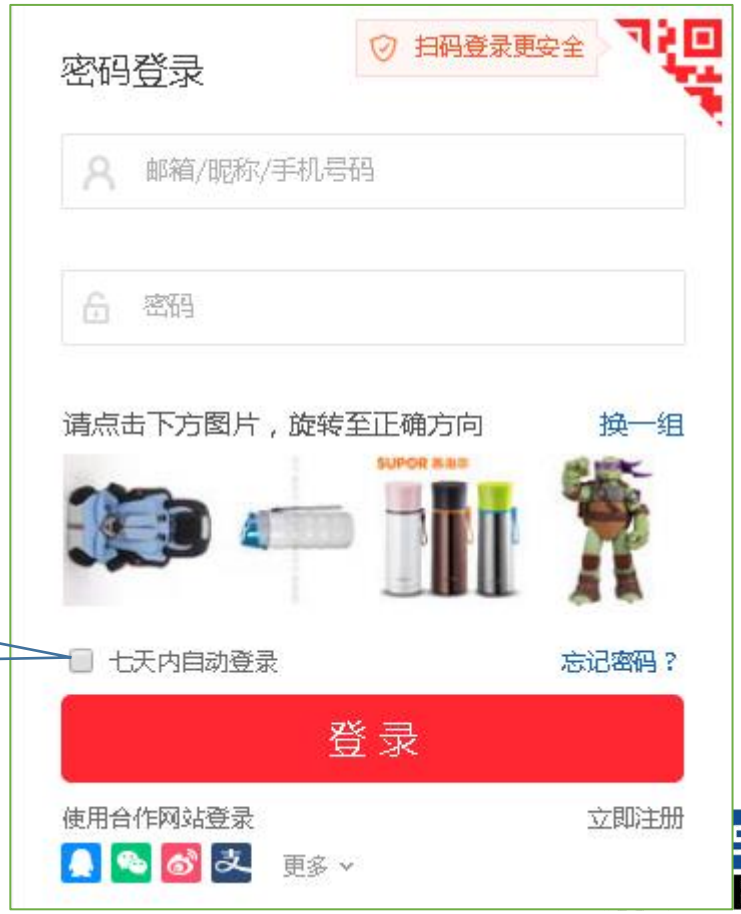
知识点1-会话的概念与作用-2

- HTTP协议是无状态的，也就是说，一次请求结束后，HTTP协议就不再记录相关信息；
- 而实际应用中，却常常需要记住一些状态信息；

要下载文档，要求必须先登录，称为访问控制。

会话跟踪技术就能够实现这样的功能：能够跟踪客户端与服务器端的交互，保存和记忆相关的信息，保存请求的状态信息。

一定时间内不再需要重新登录，称为简化登录。



知识点1-会话的概念与作用-2

- 目前唯一的方法：

现在的应用环境中，能够采用的方法是：

每次向服务器发送请求时

都

主动携带身份令牌信息

知识点2-现行常用的会话跟踪技术

- 常用的会话跟踪技术有四种
  - URL方式：需要保存的信息直接追加到URL后，例如：
    http://127.0.0.1:8080/chapter03/viewList?pageNo=12
  - 隐藏域方式：可以使用表单中的隐藏域保存相关信息， 例如：

```
<input type="hidden" name="status" value="true">
```

  - Cookie方式：将状态信息保存到客户端，服务器能够获得相关信息进行分析，从而生成对客户端的响应；例如简化登录功能就可以使用Cookie实现；
  - Session方式：将状态信息保存到服务器的会话对象中，通过唯一标记的ID值与客户端进行绑定使用；例如访问控制功能就可以使用Session实现；

本节总结提问【会话跟踪概述】

- 什么是会话?
- 会话跟踪有什么作用?
- 有几种常用的会话跟踪技术?

本节总结【会话跟踪概述】

- Web应用中，浏览器与服务器之间一次连续的通讯过程，称为一次会话；
- 会话跟踪能够保存状态信息，解决HTTP协议无状态特性的弊端；
- 目前常用的会话跟踪机制有：URL、隐藏域、Cookie、Session；

第2节【Cookie】

- 知识点1： Cookie的功能与特点
- 知识点2： Cookie的域及最大生命时间
- 知识点3：在Servlet中创建Cookie、设置Cookie属性
- 知识点4：在响应中设置Cookie信息
- 知识点5：获取请求中的Cookie信息

知识点1-Cookie的功能与特点

- Cookie是一段保存在客户端的小文本；能够用来将用户活动过程中的状态信息保存到客户端，服务器可以获得该信息以便进行处理，跟踪到用户的状态；
- 不同的浏览器有不同的查看方式；以Chrome浏览器为例，查看当前页面cookie 的方法:点进去设置——>高级——>网站设置——>权限——>cookie和网站数据,就可以看到所有的cookie信息了

知识点2-Cookie的域及最大生命时间

- Cookie包含一系列的属性，如下图所示：

- name：cookie的名字，每个cookie都有一个名字；
- content：cookie的值，与名字一起作为键值对形式存在；
- domain：域，该cookie的域名，例如左图中是163.com，说明当前cookie来自163.com;
- path：路径，访问163.com下该路径时，当前cookie将被发送；
- created：cookie被创建的时间；
- Expired：cookie失效的时间；
- **最大生命时间**：失效时间和创建时间的时间差，就是cookie的最大生命时间，超过该时间，cookie将失效，不再被发送到相应的域地址；

知识点3-在Servlet中创建Cookie、设置Cookie属性-1

- Servlet规范中定了Cookie类，创建该类对象就可以创建Cookie，并可以调用其中方法为Cookie设置属性；

方法声明	方法描述
Cookie(java.lang.String name, java.lang.String value)	创建Cookie对象，指定名字和对应的值；
void setMaxAge(int expiry)	设置最大生命时间（秒），如果不设置，当前浏览器关闭，cookie即失效；
void setValue(java.lang.String newValue)	设置Cookie的值；
setDomain(java.lang.String domain)	设置cookie的域名；

知识点3-在Servlet中创建Cookie、设置Cookie属性-2

- Servlet规范中定了Cookie类，创建该类对象就可以创建Cookie，并可以调用其中方法为Cookie设置属性；

```
<%
 //          创建cookie对象
             Cookie [] c = request.getCookies();
             if(c== null){
                          String i = "2";
                          Cookie ck = new Cookie("count",i);
                          //设置cookie的生命时间，24小时内有效
                          ck.setMaxAge(60*60*24);
}
```

- 思考：创建好Cookie对象后，就能存储到客户端吗？
- 答案是：No

知识点4-在响应中设置Cookie信息

- 要将Cookie保存到客户端，就要将其添加到响应对象才可以；
- 响应接口中定义了设置Cookie的方法：

方法声明	方法描述
void addCookie(Cookie cookie)	将Cookie对象保存到相应的响应对象中；

```
//                  将Cookie保存到响应中
                    response.addCookie(ck);
```

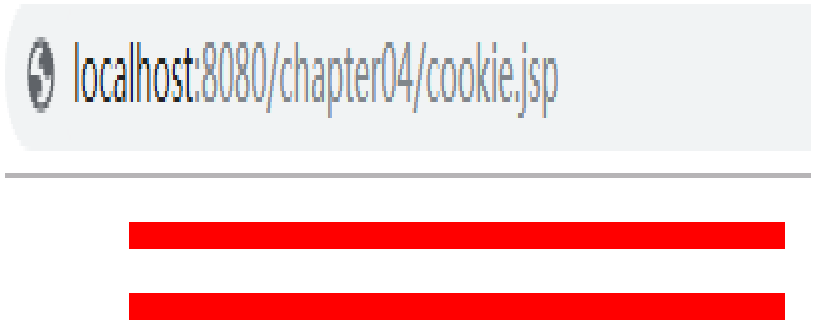
课堂案例：
cookie.jsp

```
<%
        Cookie [] c = request.getCookies();
        if(c== null){
                String i = "2";
                Cookie ck = new Cookie("count",i);
                ck.setMaxAge(60*60*24);
                response.addCookie(ck);
        }else{
                for(int i = 0 ; i < c.length;i++){
                        String s = c[i].getName();
                        if(s.equals("count")){
    out.println("欢迎您访问我们的网站，这是您第"+c[i].getValue()+"次访问，欢迎以后常来！！！");
        int j = Integer.parseInt(c[i].getValue())+1;
                                Cookie ck = new Cookie("count", new Integer(j).toString());
                                // ck.setMaxAge(0);
                                ck.setMaxAge(60*60*24);
                                response.addCookie(ck);
                        }}
}%>
```

知识点4-在响应中设置Cookie信息-3

课堂案例：
cookie.jsp

- 在浏览器中访问
- http://localhost:8080/chapter04/cookie.jsp
- 可以查看到locahost保存了两个Cookie信息到客户端：



第一次访问为空，之后依次累加

知识点5-获取请求中的Cookie信息-1

- 当访问相同域及路径时，没有超过有效时间的cookie将自动通过请求被发送到网站；
- Servlet规范中的请求接口定义了获取Cookie对象的方法：

方法声明	方法描述
Cookie[] getCookies()	获取请求中的所有Cookie对象，返回数组；

课堂案例：
getCookie.jsp

```
<form name="form" action="GetCookieServlet" method="post">
  <table>

                    <tr><td>Cookie属性名为：</td>
                        <td><input type = "text" name = "name"/></td>
                    </tr>
                    <tr><td>Cookie属性值为：</td>
                                <td><input type = "text" name = "value"/>      </td>
                    </tr>
            <tr><td><input type = "submit" value = "提交"/></td>
             <td><input type = "reset" value = "重置"/></td></tr>
  </table>
</form>
```

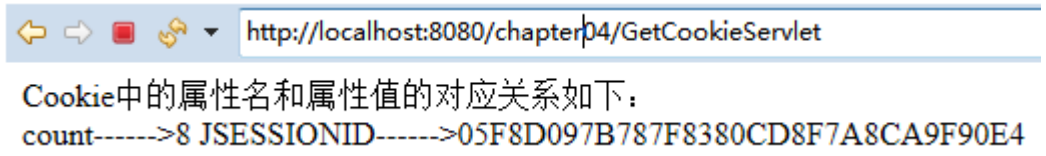
课堂案例：
GetCookieServlet.java

```
package com.chinasofti.chapter04.section02;
@WebServlet("/GetCookieServlet")
public class GetCookieServlet extends HttpServlet {
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
                request.setCharacterEncoding("utf-8");
                response.setContentType("text/html;charset= utf-8");
                PrintWriter out = response.getWriter();
                String name = request.getParameter("name");String value = request.getParameter("value");
                System.out.println("cookieName="+ name);System.out.println("cookieValue="+value);
                if(name!=null) {
                //创建cookie对象
                Cookie c = new Cookie(name,value);response.addCookie(c);
                }else {System.out.println("Cookie为空！ ！ ！ ");}
                Cookie [] cookies = request.getCookies();
                out.println("Cookie中的属性名和属性值的对应关系如下： ");out.println("<br>");
                if(cookies!=null) {
                        for(Cookie cookie:cookies){
                                System.out.println("@@@@@@@@@@@@@@@");
                                out.println(cookie.getName()+"------>"+cookie.getValue());}}}
```

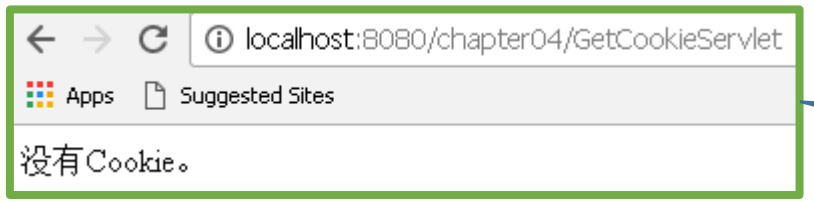
知识点5-获取请求中的Cookie信息-2

课堂案例：
GetCookieServlet.java

- http://localhost:8080/chapter04/getCookie.jsp，保证Cookie信息保存到客户端；
- 表单提交到GetCookieServlet，获取Cookie并进行显示；



- 如果将时间调到24小时后，再次访问GetCookieServlet，发现Cookie已经不存在：



由于GetCookieServlet中Cookie的生命时间设置为24小时，所以24小时后就失效。

如果没有使用setMaxAge方法设置最大生命时间的Cookie，则浏览器关闭就失效；

本节总结提问【Cookie】

- Cookie有什么作用?
- Servlet规范中的Cookie类有哪些方法?
- 如何将Cookie保存到客户端?
- 如何获取Cookie?

本节总结【Cookie】

- Cookie是保存到客户端的小文本，可以用来跟踪用户状态；
- Cookie类可以创建Cookie对象，能够对Cookie设置最大生命时间、domain值、value值等；
- 响应接口中的addCookie方法可以添加Cookie到响应，从而保存到客户端；
- 请求接口中的getCookies方法可以获取当前请求中所有Cookie对象；
- 设置了最大生命时间的Cookie可以在有效时间内使用，没有设置的只是临时的，浏览器关闭即失效；

第3节【Session】

- 知识点1：Session简介
- 知识点2：Session使用方法
- 知识点3：HttpSession对象的获取
- 知识点4：Session常用方法

知识点1：Session简介-1

- 事实上，在应用系统中直接使用持久化的Cookie来保持用户的会话信息在很多情况下是不可取的，因为：

总有无聊的人对你的个人信息感兴趣

知识点1：Session简介-1

- 因此WEB服务器均提供了更灵活的使用方法：

应用服务器的
常见策略：

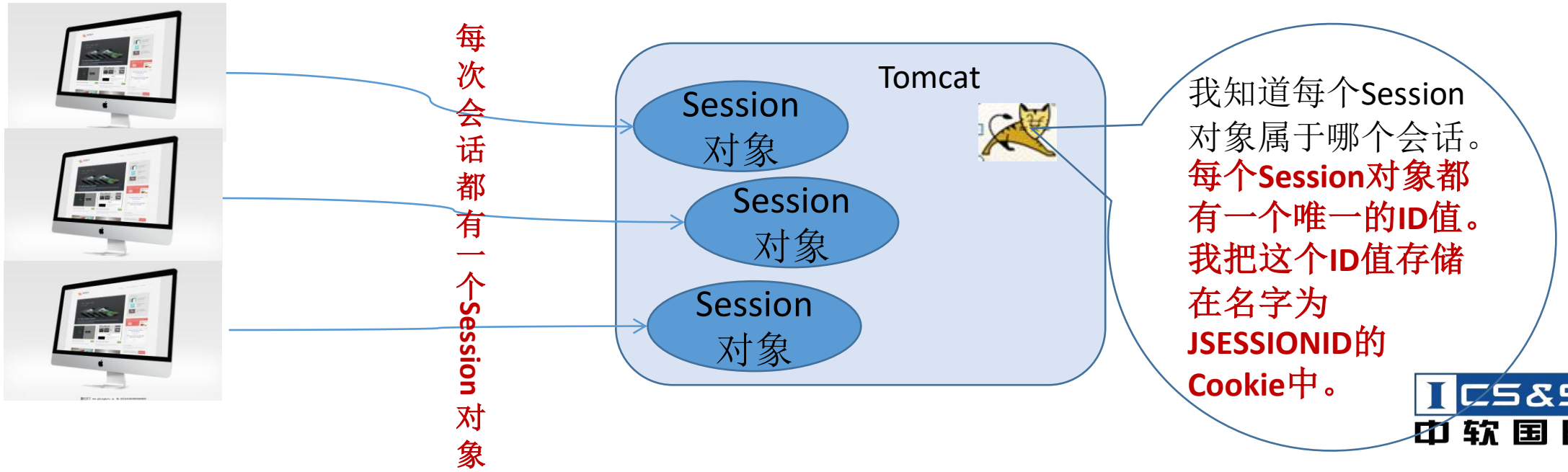
利用会话型Cookies分发令牌，构建Session

-会话型Cookies不会保存本地物理数据，Session只分发唯一的但无意义的令牌信息

身份数据保存在服务器内存中

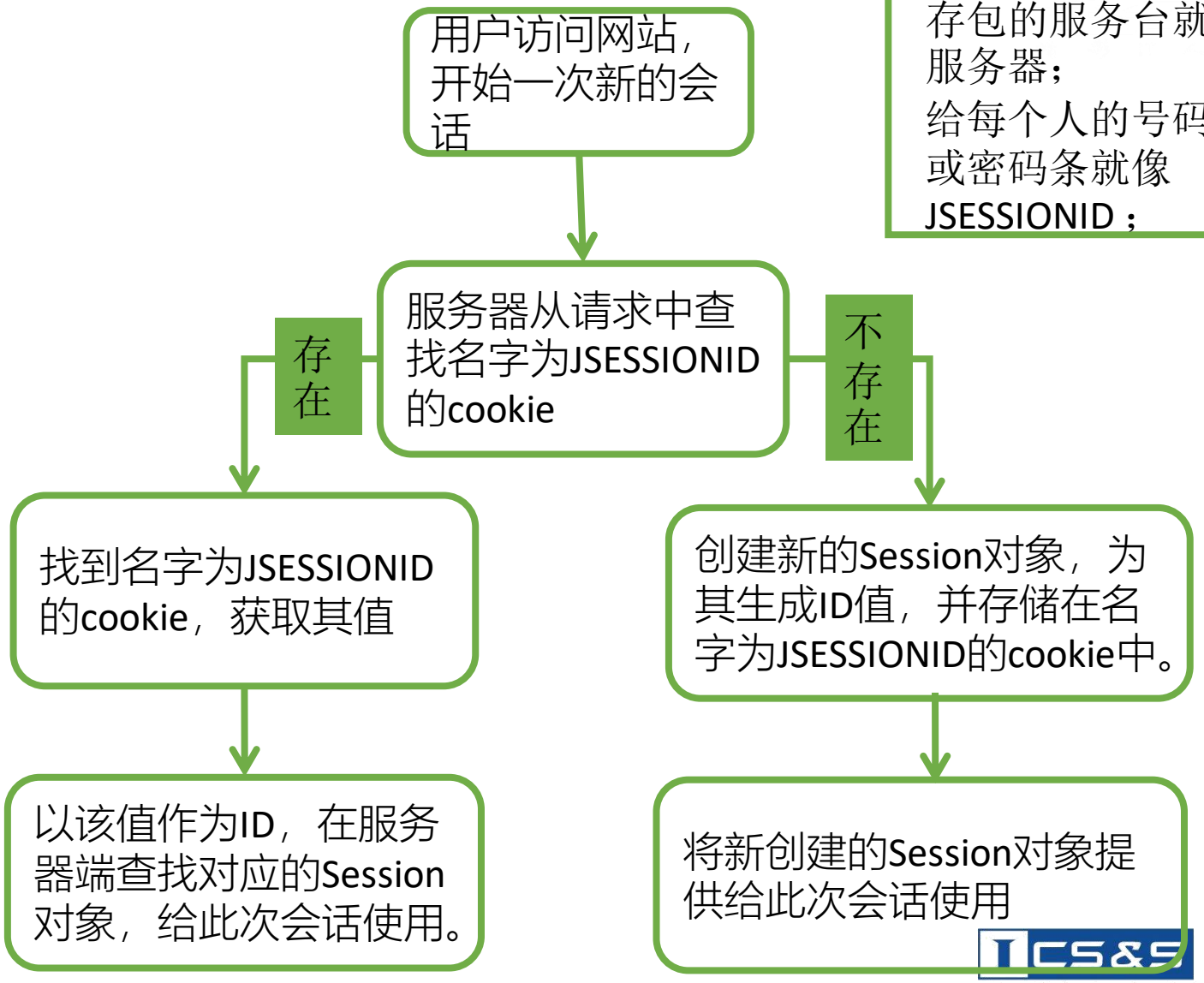
知识点1：Session简介-1

- Session是会话跟踪的另一种实现手段；
- Session是存储在服务器上的对象，该对象由服务器创建并维护；
- 服务器为客户端与服务器的每一次会话过程都创建并维护一个Session对象；每个服务器对Session的创建和维护的底层实现有所区别；



知识点1：Session简介-2

- 每种服务器对Session的创建和维护底层实现有所区别；
- Tomcat使用Cookie来维护Session对象的ID值；该Cookie名字为JSESSIONID ；
- 当客户端开始一次会话过程时，以Tomcat为例，简略步骤如图：



知识点2：Session使用方法-1

- Session 是用于保持状态的基于 Web服务器的方法。Session 允许通过将对象存储在 Web服务器的内存中在整个用户会话过程中保持任何对象。
- Session 通常用于执行以下操作
- 存储需要在整个用户会话过程中保持其状态的信息，例如登录信息或用户浏览 Web 应用程序时需要的其它信息。
- 存储只需要在页面重新加载过程中或按功能分组的一组页之间保持其状态的对象。
- Session 的作用就是它在 Web服务器上保持用户的状态信息供在任何时间从任何设备上的页面进行访问。因为浏览器不需要存储任何这种信息，所以可以使用任何浏览器，即使是像 Pad 或手机这样的浏览器设备。

知识点2：Session使用方法-2

- 持久性方法的限制
- 随着越来越多用户登录，Session 所需要的服务器内存量也会不断增加。
- 访问 Web应用程序的每个用户都生成一个单独的 Session 对象。每个 Session 对象的持续时间是用户访问的时间加上不活动的时间。
- 如果每个 Session 中保持许多对象，并且许多用户同时使用 Web应用程序(创建许多 Session)，则用于 Session 持久性的服务器内存量可能会很大，从而影响了可伸缩性。

知识点2：Session使用方法-3

- 实现自定义session的基本步骤：
  - Session的实现仍然依靠临时态的Cookie，因此需要定义一个属于自定义session体系的Cookie名称（如JavaEE应用服务器中常用的JSESSIONID）
  - 在服务端创建一个Map，用于存放所有的用户会话
  - 当用户访问系统时，检查请求中是否存在以自定义session体系Cookie名称命名的Cookie信息
    - 如果没有，说明是一个新用户，则为该用户生成一个唯一的sessionId字符串，并在响应中添加该Cookie值（注意不要提供Cookie的生效时间，那么该Cookie即为瞬态的，只在浏览器当前进程中生效）
    - 如果有则查看Map中是否存在该会话，有即获取会话信息，没有说明该会话已经超时，为用户构建一个新的会话
  - 一个用户的会话对象实质上也是一个Map，可以通过键值对的形式存放、获取会话变量

知识点3：HttpSession的获取-1

- Servlet规范中定义了HttpSession接口，用来实现Session技术；
- 要使用HttpSession，首先要获取其对象；请求接口中定义了获取HttpSession对象的方法：

方法声明	方法描述
HttpSession getSession()	获取与当前请求相关的Session对象，如果不存在，创建一个新的；
HttpSession getSession(boolean create)	如果create为true，则与getSession()方法相同；如果create是false，则如果不存在，返回null;

知识点3：HttpSession的获取-2

课堂案例：
TestGetSession.java

- 编写TestGetSession，先获取名字为JSESSIONID的cookie，再获取HttpSession对象，并获取其ID值，比较二者关系；

```
//    获取jsessionid，并输出其值
      Cookie[] cookies=request.getCookies();
      boolean flag=false;
      if(cookies!=null){
      for(Cookie c:cookies){
            if(c.getName().equals("JSESSIONID")){
                  out.println("JSESSIONID="+c.getValue());
                  flag=true;
                  break;          }      }      }
//    如果不存在jsessionid，则打印提示
      if(flag==false){
            out.println("当前请求中没有名字为JSESSIONID的cookie");
      }
 //  获取当前Session对象
      HttpSession session=request.getSession();
//    获取当前Session对象的ID
      String id=session.getId();
      out.println("当前会话的ID是："+id);
```

知识点3：HttpSession的获取-2

课堂案例：
TestGetSession.java

- 在浏览器中访问TestGetSession，结果如下：

```
当前请求中没有名字为JSESSIONID的cookie
当前会话的ID是：1ACDCD12721AA476E8D7EDE1A3D910BD
```

可见：第一次访问，没有创建过HttpSession对象，所以也不存在名字为JSESSIONID的Cookie；

- 同一个浏览器中，再次访问TestGetSession，结果如下：

```
JSESSIONID=1ACDCD12721AA476E8D7EDE1A3D910BD
当前会话的ID是：1ACDCD12721AA476E8D7EDE1A3D910BD
```

可见：同一个会话中，只创建唯一一个HttpSession对象，而且ID值确实存储在名字为JSESSIONID的Cookie中；

- 再启动新的浏览器进行访问，结果如下：

```
当前请求中没有名字为JSESSIONID的cookie
当前会话的ID是：50EAAF970E39CA18CB85D983F1866A5F
```

可见：启动新的浏览器，即开始一个新的会话，将创建新的HttpSession对象，有不同的ID值；

知识点4：Session的方法-1

- 与前面学习过的请求属性类似，会话也可以添加、修改、删除属性；
- HttpSession接口中提供了与属性有关的方法；

方法声明	方法描述
void setAttribute(java.lang.String name, java.lang.Object o)	将任意类型对象设置为会话的属性，指定一个名字；
java.lang.Object getAttribute(java.lang.String name)	通过属性的名字，获取属性的值；
void removeAttribute(java.lang.String name)	通过属性的名字，删除属性；

- 有了会话属性，就可以在会话范围内共享对象；

知识点4：Session方法-2

课堂案例：
TestSessionAttr.java
getSessionAttr.jsp

- 在Servlet中添加Session的属性，响应重定向到JSP进行显示：

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
//                  获得HttpSession对象
                    HttpSession session=request.getSession();
//                  设置Session属性
                    session.setAttribute("username", "wangxh");
//                  响应重定向到JSP，到JSP中获取会话属性
                    response.sendRedirect("getSessionAttr.jsp");
}
```

思考一下：如果是请求中的属性，响应重定向到JSP中还能获取么？

- 在JSP中进行显示：

```
<body>
   会话属性：<%=session.getAttribute("username")%>
</body>
```

可见：会话对象中可以添加属性，在需要的时候获取使用即可；

知识点4：Session方法-3

- 为什么要让Session失效?
  - 会话对象是存储在服务器端的对象，一直存在需要占用一定的服务器资源；
  - 会话中往往保存着用户的一些数据，如果一直有效，存在一定安全隐患。
- Session失效的方法
  - 服务器都有默认的会话失效时间，Tomcat默认是30分钟；
  - 可以在web.xml中配置失效时间，例如：配置失效时间是50分钟；

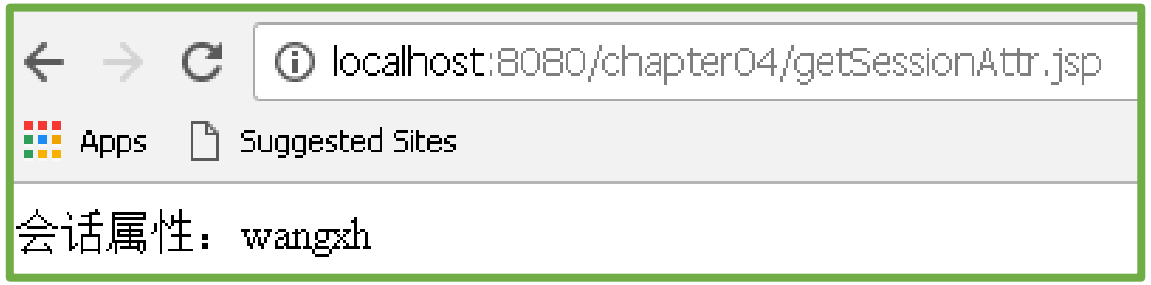
```
<session-config>
  <session-timeout>50</session-timeout>
</session-config>
```

  - 调用HttpSession接口中的两个方法，可以对指定的会话对象进行销毁；

方法声明	方法描述
void setMaxInactiveInterval(int interval)	为特定的会话对象设定不活动时间，超过这个时间内没有被访问使用，容器自动销毁该会话对象；
void invalidate()	立刻销毁调用该方法的会话对象，并把所有绑定到该会话的对象解除绑定；

知识点4：Session-4

- 访问TestSessionAttr，再访问getSessionAttr.jsp，可以在JSP中显示会话属性：



- 修改系统时间，30分钟后，再访问getSessionAttr.jsp，可见属性为空，说明会话被销毁；

提示：可以在web.xml中配置新的失效时间进行测试；也可以对Session对象设置不活动时间进行测试；

本节总结提问【Session】

- HttpSession对象如何获取?
- HttpSession如何存取属性?和请求属性有什么区别?
- 会话失效有几种方式?

本节总结【Session】

- 请求接口中定义了获得HttpSession对象的方法getSession;
- HttpSession接口中定义了和属性有关的方法，setAtrribute、getAttribute、removeAttribute
- 会话失效有四种情况：超过服务器默认的有效时间、超过web.xml配置的有效时间、超过使用setMaxInterval方法设定的时间、调用了invalidate方法;

本章总结

- 会话跟踪技术是实际Web应用开发中常用到的技术，主要包括URL、隐藏域、Cookie、Session；
- Cookie用来把简单信息保存到客户端，以便跟踪用户状态；Servlet规范中定义了Cookie类，实现Cookie技术；
- Session是存储在服务器端的对象，一次会话过程中有一个唯一的Session对象；
- Servlet规范中定义了HttpSession接口，能够实现Session技术；
- HttpSession对象中可以添加属性，用来在会话范围共享数据；

本章作业

- 作业:

  1）创建cookie对象，设置最大生命时间一小时

  2）应用cookie技术实现登录某网站统计登录次数

  3）创建session对象，查看API实现其常用方法

  难度：中