**会话管理**

# 1 会话的概念

会话:用户打开浏览器，浏览页面，直到关闭浏览器的过程称为一次会话。

**为什么要学习会话技术？**

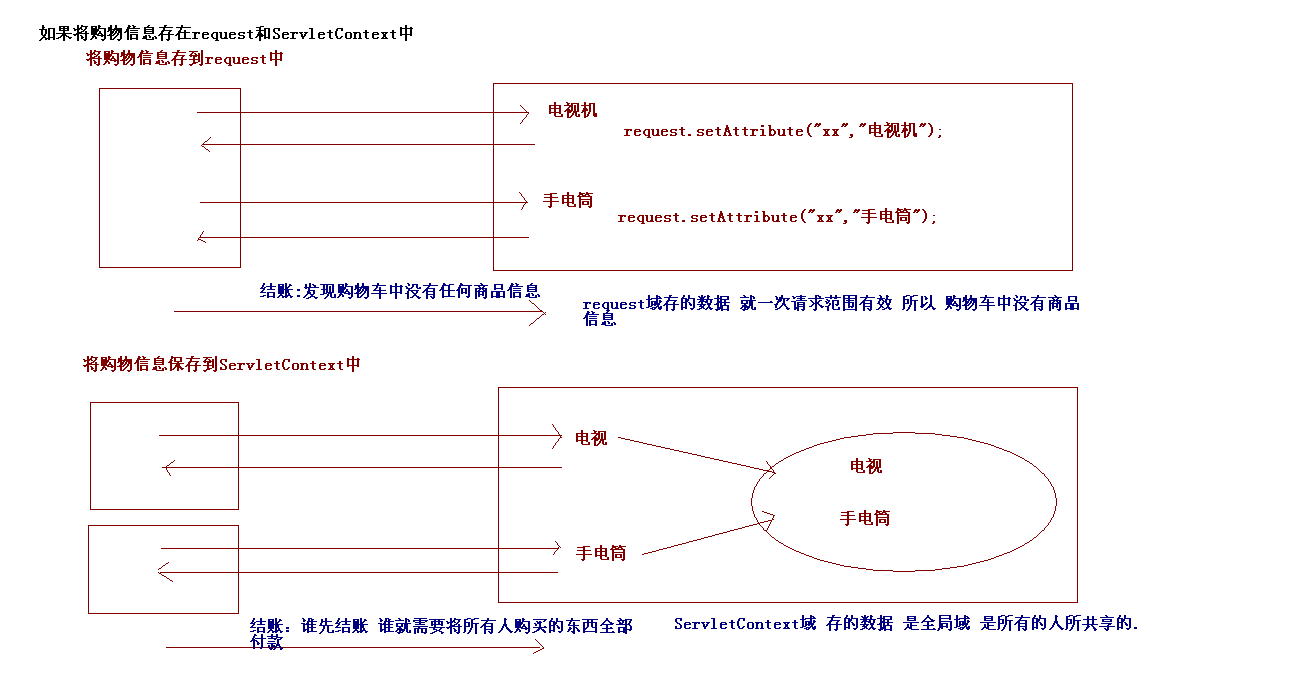
http协议是一个无状态协议，服务器端无法记录客户端浏览器身份信息。

例如：购物网站，购物车（需要记录用户购买商品的信息）。

如何做到？ request？ 无法实现，会发现结账时，没有任何商品信息。

ServletContext？ 无法实现，结账时会发现，所有人的商品

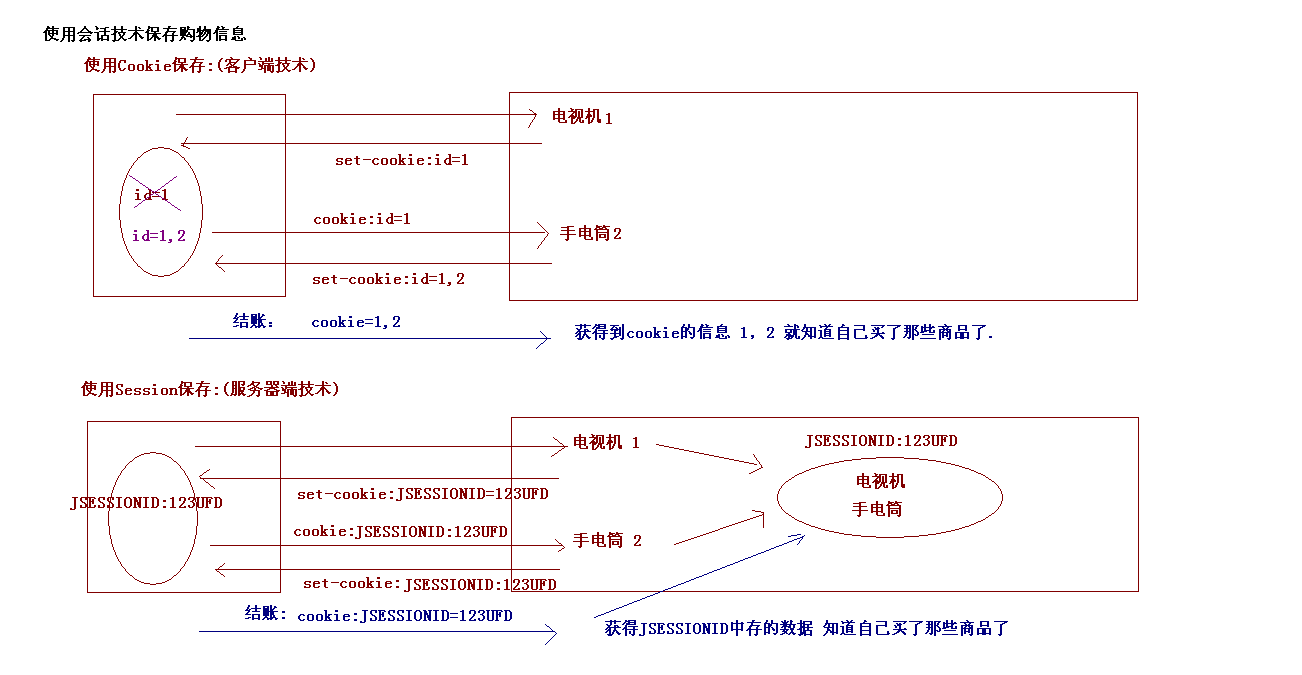
信息都在购物车中。



因此，需要使用会话跟踪技术解决。

会话跟踪技术包括： **Cookie（客户端技术）**

**Session（服务器端技术）**

****

# 2 Cookie

**javax.servlet.http.Cookie**

### Cookie简述

Cookie翻译后 小甜点、小饼干

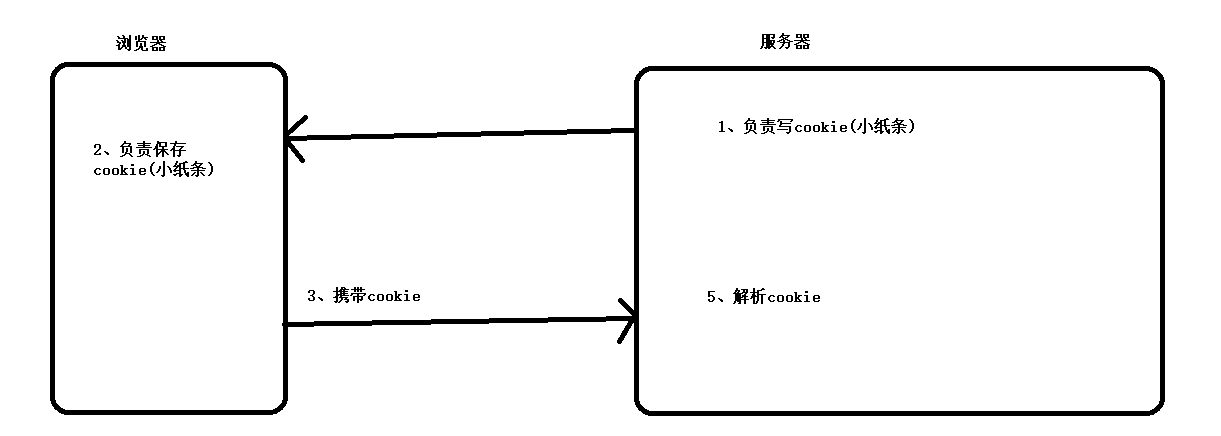
Cookie 是浏览器和服务之间传递的一张小纸条

服务器负责：

1. 负责写cookie
2. 负责解析浏览器传过来的cookie

浏览器负责：

1. 负责保存cookie
2. 负责将cookie发送给服务器



## 2.1 Cookie常用方法

1 构造方法： Cookie c = new Cookie(“”,“”)；

创建Cookie时， name和value都是字符串。**注意：不能用中文字符串。**

2 getName() 获得Cookie的name

3 getValue() 获得Cookie的value **注意：Cookie没有提供根据name获取value的方法**

4 setMaxAge(int 秒) 设置Cookie的存活时间

5 setPath() 设置Cookie的有效路径

浏览器将cookie保存硬盘的路径，是各个浏览器自己制定，其他无法进行操作的。

有效路径：浏览器发送请求时，cookie的代入路径。

请求路径 等于或者包含cookie的有效路径，该cookie会在请求时被代入。

**请求路径是从资源位置开始进行判断**

|  |  |
| --- | --- |
| 方法名 | 描述 |
| **setPath**(java.lang.String uri) | 设置有效路径。  设置cookie允许被访问的路径。设置的路径，以及子路径都被允许访问。   * 例如：setPath("/web/a/b");   http://localhost:8080/web/a/b/oneServlet，可访问(当前路径)  http://localhost:8080/web/a/b/c/oneServlet，可访问(子路径)  http://localhost:8080/web/a/c/oneServlet，不允许访问(无关路径)   * 常见设置：setPath(“/”) ，当前tomcat下的所有的web项目都可以访问 |

7 获得客户端带过来的Cookie

Cookie[] cs = request.getCookies()

8 服务器端把Cookie带到客户端

**response.addCookie(cookie);**

## 2.2 Cookie案例

### 2.2.1 利用Cookie记录客户端是否是第一次访问

**需求：创建一个FirstCookieServlet 客户端访问这个Servlet**

**如果是第一次访问这个Servlet 显示 欢迎第一次访问**

**如果不是第一次访问这个Servlet 显示 欢迎回来**

|  |
| --- |
| public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {    // 判断浏览器端是否带有cookie  // 如果没有，那么是第一次访问。显示第一次访问，创建一个cookie 返回给客户端  // 如果有cookie 则不是第一次访问 显示欢迎回来    String str = "欢迎第一访问！！！";  boolean flag = false; // 假设找cookie时找不到  Cookie[] cs = request.getCookies(); //得到浏览器端带过来的cookie  if(cs!=null){  //当cs是空时，肯定是第一次访问。当cs不是空时，分成两种情况  // 遍历这个cookie数组，查找当前servlet响应个客户端的那个cookie  // 如果没找到 是第一次访问  // 如果找到了 则不是第一次访问  for(Cookie c:cs){  if("visited".equals(c.getName())){ // 找到了当前servlet响应给客户端的cookie  flag = true;  break;  }  }  }  if(flag){  str = "欢迎回来";  }else{  //创建Cookie Cookie(String name, String value)  Cookie cookie = new Cookie("visited","true");  cookie.setMaxAge(2); //设置Cookie的存活时间  // cookie.setPath("/"); //设置Cookie的有效路径  // 把Cookie响应给客户端  response.addCookie(cookie);  }  response.setContentType("text/html;charset=utf-8");  PrintWriter out = response.getWriter();  out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");  out.println("<HTML>");  out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");  out.println(" <BODY>");  out.println("<h1>"+str+"</h1>");  out.println(" </BODY>");  out.println("</HTML>");  out.flush();  out.close();  } |

### 2.2.2 利用Cookie记录客户端上次访问时间

1.接收客户端带过来Cookie的信息

2.从数组中查找指定名称Cookie，如果找不到(第一次访问)，输出一个欢迎

如果找到了(不是第一次)，取出上次的访问时间，显示到页面

3.记录系统当前时间

4.利用Cookie回写当前时间

|  |
| --- |
| **public** **void** doPost(HttpServletRequest request, HttpServletResponse response)  **throws** ServletException, IOException {    // 1 获得客户端带过来的Cookie[]    // 2 如果没有找到 ，显示欢迎光临！！！（说明是第一次访问） ----> 做一个Cookie 记录本次访问时间，带给客户端  // 3 如果找到了，显示上次访问的时间    String str = "欢迎光临...";    Cookie[] cs = request.getCookies();    **if**(cs!=**null**){    **for**(Cookie c:cs){    **if**("visitedTime".equals(c.getName())){    str = "上次访问时间为："+c.getValue();    **break**;  }  }  }      String date = DateUtil.*dateToString*(**new** Date());    Cookie c = **new** Cookie("visitedTime",date);    c.setMaxAge(10000);  c.setPath("/");  response.addCookie(c);  response.setContentType("text/html;charset=utf-8");  PrintWriter out = response.getWriter();  out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");  out.println("<HTML>");  out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");  out.println(" <BODY>");  out.println("<h1>"+str+"</h1>");  out.println(" </BODY>");  out.println("</HTML>");  out.flush();  out.close();  } |

### 2.2.3 利用Cookie记录客户端访问次数

|  |
| --- |
| public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  boolean flag = false;  Cookie[] cs = request.getCookies();  Cookie cc = null;  if (cs != null) {  for (Cookie c : cs) {  if ("visited".equals(c.getName())) { // 找到了  cc = c;  flag = true;  }  }  }  int num = 1;  if (flag) {  num = Integer.parseInt(cc.getValue()) + 1;  }  Cookie c = new Cookie("visited", num + "");  response.addCookie(c);  response.setContentType("text/html;charset=utf-8");  PrintWriter out = response.getWriter();  out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");  out.println("<HTML>");  out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");  out.println(" <BODY>");  out.println("第" + num + "次访问");  out.println(" </BODY>");  out.println("</HTML>");  out.flush();  out.close();  } |

## 2.3 Cookie的分类

1 瞬时Cookie 存在浏览器的内存中

2 持久化Cookie 存在一个本地的文件中

**如何设置成持久化Cookie？**

setMaxAge() 将瞬时Cookie设置成持久化Cookie，存在磁盘的文件中。

**查看浏览器保存Cookie的文件**

**IE：**

C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Cookies

**FireFox：**

工具---选项---隐私---使用自定义历史设置里面查看

## 2.4 Cookie的细节

1、一个WEB站点可以给一个WEB浏览器发送多个Cookie，一个WEB浏览器也可以存储多个WEB站点提供的Cookie。

2、浏览器一般只允许存放300个Cookie，每个站点最多存放20个Cookie（新版本的浏览器有所提升），每个Cookie的大小限制为4KB。

## 2.5 Cookie案例2

**1、利用Cookie记录用户的浏览记录:(网站的真实案例)**

产品列表页面

|  |
| --- |
| <body>  <a href="ViewServlet.do?id=1"><img src="img/1.jpg" />手电筒</a>  <a href="ViewServlet.do?id=2"><img src="img/2.jpg" />手机</a>  <a href="ViewServlet.do?id=3"><img src="img/3.jpg" />电视</a>  <a href="ViewServlet.do?id=4"><img src="img/4.jpg" />冰箱</a>  <a href="ViewServlet.do?id=5"><img src="img/5.jpg" />手表</a>  <a href="ViewServlet.do?id=6"><img src="img/6.jpg" />电脑</a>  <hr />  浏览过的物品：<a href="ClearServlet.do">清空浏览记录</a>  <%  Cookie[] cs = request.getCookies();  Cookie c = CookieUtil.findCookieByName("product", cs);  if(c!=null){  String value = c.getValue();  String[] ss = value.split(",");  for(String s:ss){  %>  ssss  <img src="img/<%=s %>.jpg" width="100" height="100" />  <%  }  }  %>  </body> |

Servlet

|  |
| --- |
| public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  // 获得客户端的物品的id  String id = request.getParameter("id");  /\*  \* 如果使用了数据库 那么要根据id查询出这个物品的对象  \*  \* 放入request中  \*  \* \*/  Cookie[] cs = request.getCookies();  if(cs==null){ // 第一次访问  Cookie c = new Cookie("product",id);  c.setMaxAge(10000);  c.setPath("/");  response.addCookie(c);  }else{ //不是第一次访问  Cookie cookie = CookieUtil.findCookieByName("product", cs);  if(cookie==null){ // 第一次访问  Cookie c = new Cookie("product",id);  c.setMaxAge(10000);  c.setPath("/");  response.addCookie(c);  }else{ //不是第一次访问  //判断是否点过该商品  // 1,2,3,5  String value = cookie.getValue();  String[] ss = value.split(",");  boolean flag = this.checkId(ss, id);  if(!flag){ //没有点过该物品  Cookie c = new Cookie("product",value+","+id);  c.setMaxAge(10000);  c.setPath("/");  response.addCookie(c);  }  }  }  request.setAttribute("id",id);  request.getRequestDispatcher("viewdetail.jsp").forward(request, response);  }  public boolean checkId(String[] ss,String id){// 判断是否点过某个商品  boolean flag = false;  for(String s:ss){  if(s.equals(id)){  flag = true;  break;  }  }  return flag;  } |

CookieUtil 查找指定Cookie的工具类

|  |
| --- |
| public class CookieUtil {  public static Cookie findCookieByName(String name, Cookie[] cs) {  if (cs != null) {  for (Cookie c : cs) {  if (name.equals(c.getName())) {  return c;  break;  }  }  }  return null;  }  } |

清除访问记录的Servlet

|  |
| --- |
| public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  //删除浏览记录  // 找到浏览记录的Cookie  // setMaxAge(0)  Cookie[] cs = request.getCookies();  Cookie c = CookieUtil.findCookieByName("product", cs);  if(c!=null){  c.setPath(“/”); //必须有相同的有效路径才能清除  c.setMaxAge(0);  response.addCookie(c);  }  } |

**2 登录时记录登录名**

登录页面：

|  |
| --- |
| <body>  <%  Cookie[] cs = request.getCookies();  Cookie c = CookieUtil.findCookieByName("loginName", cs);  String loginName = null;  if(c!=null){  loginName = URLDecoder.decode(c.getValue(), "UTF-8");    }  %>  <form action="LoginServlet.do" method="post">  loginname:  <input type="text" name="loginName" value="<%=loginName!=null?loginName:"" %>" /><br>  password:<input type="password" name="pwd" /><br />  <select name="remember">  <option value="0">不记录用户名</option>  <option value="7">一周</option>  <option value="30">一个月</option>  </select>  <input type="submit" value=" login " />  </form>  </body> |

登录Servlet

|  |
| --- |
| public void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.setCharacterEncoding("utf-8");  String loginName = request.getParameter("loginName");  String password = request.getParameter("pwd");  String remember = request.getParameter("remember");  int num = Integer.parseInt(remember);  if("张三".equals(loginName.trim()) && "123".equals(password.trim())){  //登录成功,记录登录名  if(num!=0){  String name = URLEncoder.encode(loginName, "UTF-8");// 编码  Cookie c = new Cookie("loginName",name);  c.setPath(“/”);  c.setMaxAge(num\*24\*60\*60);  response.addCookie(c);  }  }else{  response.sendRedirect("login.jsp");  }  } |

**解决Cookie中不能保存中文名值对问题:**

|  |
| --- |
| **编码：URLEncoder.encode(loginName,"utf-8");**  **解码：URLDecoder.decode("","UTF-8);** |

# 3 Session

**javax.servlet.HttpSession 服务端技术。**

会话管理已经有了Cookie，为什么还要有Session？

1 Cookie有大小和个数的限制的. Session存在服务器端没有大小和个数的限制.

2 Cookie存到客户端上 信息不安全. Session的数据相对来说安全.

**Session运行原理也是基于Cookie ：利用Cookie回写一个 JSESSIONID**

## 3.1 获取Session

方式一： HttpSession session = request.getSession()

如果session还没有创建，则创建。

如果已经创建，则使用已经创建的session。

方式二： HttpSession session = request.getSession(false);

如果session已经创建过，则使用已经创建的session。

如果session没有创建过，则返回null。

**Session的getId() 方法可以获得Session的id**

## 3.2 Session常用方法

session.setAttribute("",Object);

session.getAttribute("");

session.removeAttribute("");

## 3.3 Session案例

**1 编写Servlet,使用Session实现使用：如果是第一次访问则输出第一次访问**

**如果不是，则输出欢迎回来。**

|  |
| --- |
| public void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String str = "";  HttpSession session = request.getSession(false);  if(session==null){  str = "第一次访问";  session = request.getSession();  }else{  str = "欢迎回来";  }  response.setContentType("text/html;charset=utf-8");  PrintWriter out = response.getWriter();  out.println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\">");  out.println("<HTML>");  out.println(" <HEAD><TITLE>A Servlet</TITLE></HEAD>");  out.println(" <BODY>");    out.println(str);  out.println(" </BODY>");  out.println("</HTML>");  out.flush();  out.close();  } |

## 3.4 Session的创建和销毁

### 3.4.1 Session的创建

第一次调用getSession()是创建Session

### 3.4.2 Session的销毁

**浏览器关闭Session对象并没有销毁**

只不过是存放这个Session的Id的Cookie销毁了，导致Session对象获得不到了。

**1 关闭服务器（非正常关闭服务器）**

（如果是正常关闭服务器，Session将被序列化到服务器中）

tomcat\work\Catalina\localhost\1202Session\SESSION.ser

**2 session时间到期 session的默认存活时间是30分钟**

默认配置在tomcat的web.xml中

<session-config>

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

</session-config>

可以在自己的web.xml配置Session的过去时间，配置方式和tomcat的web.xml中配置方式相同。

**3 调用session 的 invalidate()方法**

## 3.5 Session和Cookie的关系

**Session运行原理也是基于Cookie ：利用Cookie回写一个 JSESSIONID。**

### 3.5.1 禁用浏览器端Cookie

如果客户端禁用了Cookie 那么Session将不能使用。

如何解决？（作为了解即可）

\* **手动在路径后面拼接一个**

;jsessionid=C7239BBA1C7C24E9810CA6BD3531BD4A

\* **使用response的encodeRedirectURL()方法**

String path = "/SessionDemo4Servlet.do";

String path1 = response.encodeRedirectURL(path);

response.sendRedirect(path1);

**禁用Cookie还想使用Session，那么将你网站的所有的路径 都需要进行URL的重写.**

**一般在真实应用中，当禁用了Cookie时，将不能使用Session.**

**例如购物网站，在禁用Cookie后，则无法进行购物。**

### 3.5.2 Session和Cookie的区别和联系

1.数据存储的位置不同 Cookie在客户端 Session在服务器端

2.生命周期不同

Cookie（会话Cookie浏览器关闭 持久化Cookie setMaxAge(秒数)）Session(默认30分钟，可以修改)

3.cookie只能存放字符串名值对，session可以是任意对象

4.同一个浏览器接受cookie的个数是有上限

## 3.6 Session作为作用域

Servlet的三个域对象:(存取值 存取值的时候都会有一个范围)

三个域对象的创建和销毁? 三个域对象存的值的范围是多大?

HttpServletRequest

创建和销毁?

客户端向服务器发送一次请求，服务器就会创建一个request对象.当服务器对这次请求做出响应之后 request对象就销毁了

request域的范围:

一次请求的范围.(转发一次请求)

HttpSession

创建和销毁?

第一次调用getSession()的时候

销毁:

1.服务器非正常关闭.

2.Session过期了（默认30分钟）

3.调用Session中invalidate方法

session域的范围:

一次会话的范围.

ServletContext

创建和销毁:

服务器启动的时候创建对象，服务器关闭的时候销毁.

ServletContext域的范围:

整个web项目

## 3.7 Session的典型应用

虽然Session作为作用域，范围大于request。但是，由于Session是服务器端资源，所以尽量不使用Session作用域存取数据。

Web工程HttpSession的最典型应用是，登录时，当登录成功后，把登录用户的对象保存在Session中，保证一次会话该用户的状态。退出登录时，把该用户从Session中清除。

**案例： 完成登录功能（包括验证码）**

|  |
| --- |
|  |