Author：相忠良
Email: ugoood@163.com

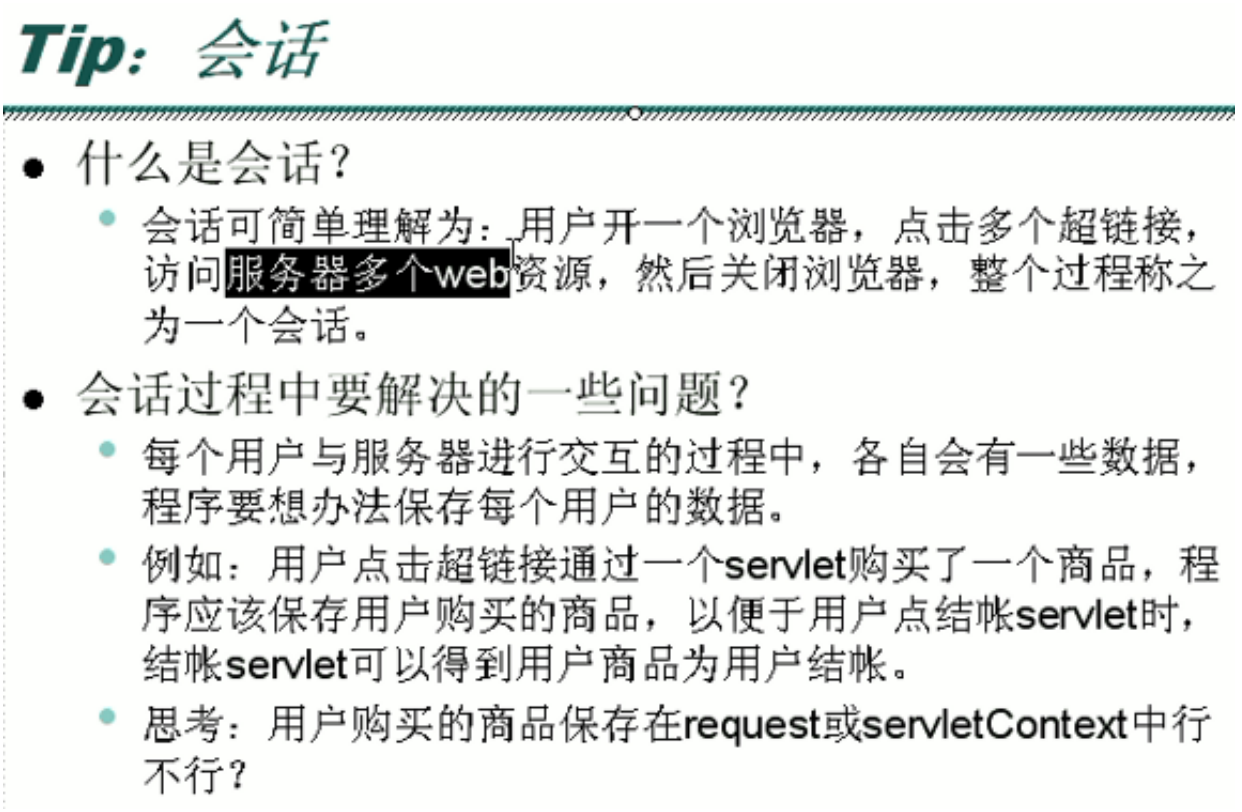
起始于：April 23, 2018
最后更新日期：April 26, 2018

声明：本笔记依据传智播客方立勋老师 **Java Web** 的授课视频内容记录而成，中间加入了自己的理解。本笔记目的是强化自己学习所用。若有疏漏或不当之处，请在评论区指出。谢谢。
涉及的图片，文档写完后，一次性更新。

# day07 Cookie 和 Session

直觉上：**Cookie** 和 **Session** 只为**1**次会话**(**简单的说：打开浏览器访问网站**--**关闭访问那个网站的浏览器，这叫会话**)**而服务。

## 1. 会话技术

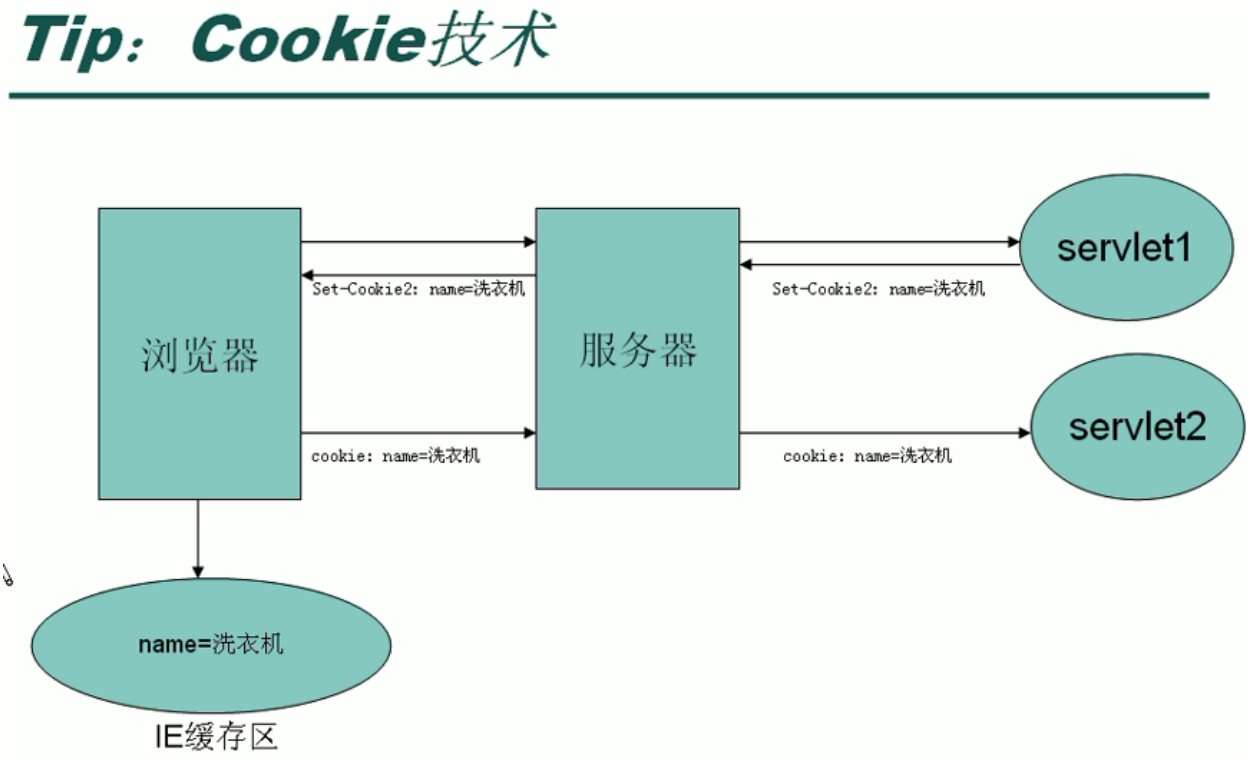



用户购买的商品保存在 request 或 servletContext 中是不行的。

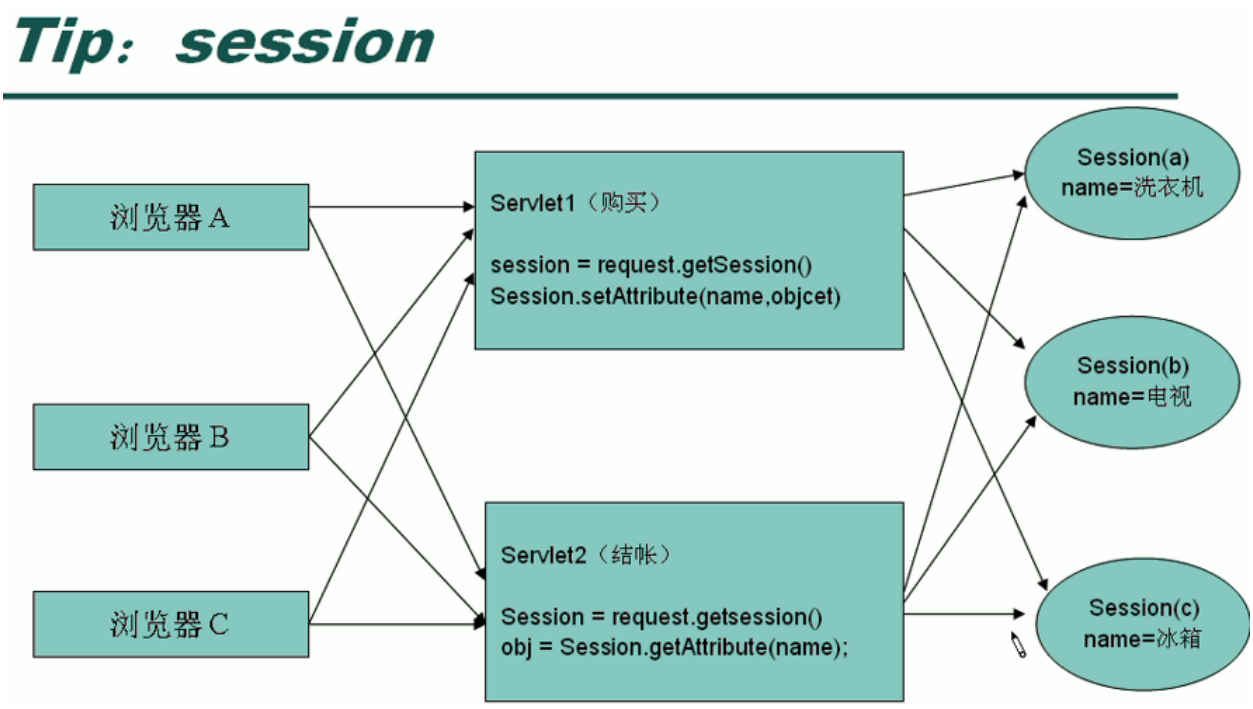
每次请求有1个request, servletContext 保存的是全局性内容，所以不可以。

day07 中讲的内容就是如何保存每位用户的数据。保存用户数据有两种方案：


Cookie 是客户端的技术：




Session 是服务器端的技术：




## 2. cookie 的方法和 cookie 案例-显示用户上次访问网站的时间

# Tip: Cookie API

- javax.servlet.http.Cookie类用于创建一个Cookie，response接口也中定义了一个addCookie方法，它用于在其响应头中增加一个相应的Set-Cookie头字段。 同样，request接口中也定义了一个getCookies方法，它用于获取客户端提交的Cookie。Cookie类的方法：
  - ✓ public Cookie(String name,String value)
  - ✓ setValue与getValue方法
  - ✓ setMaxAge与getMaxAge方法 ← 不设置的话，cookie同浏览器进程共存亡
  - ✓ setPath与getPath方法 /day06/servlet/
  - ✓ setDomain与getDomain方法 .sina.com ← 浏览器拒收，该方法没用
  - ✓ getName方法

cookie应用：显示客户上次访问时间

道理：上次访问网站时，网站写cookie送给浏览器存下来，本次访问同一个网站时，浏览器就带着cookie去，传递了上次访问时间的数据。新建 day07 的 web 工程，做如下实验：

```java
// 代表网站首页
public class CookieDemo1 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();
        out.print("您上次访问的时间是：");

        // 获取用户的时间 cookie
        Cookie[] cookies = request.getCookies();
        for (int i = 0; cookies != null && i < cookies.length; i++) {
            if (cookies[i].getName().equals("lassAccessTime")) {
                long cookieValue = Long.parseLong(cookies[i].getValue
()); // ms 得到用户的上次访问时间
                Date date = new Date(cookieValue);
                out.print(date.toLocaleString()); // 转换成本地习惯的日期
            }
        }

        // 给用户回送最新的访问时间
        Cookie cookie = new Cookie("lassAccessTime", System.currentTimeMi
llis()
                + "");
        cookie.setMaxAge(1 * 30 * 24 * 3600); // 设置有效期 秒 -- 1个月
        cookie.setPath("/day07"); // 设置该cookie适用范围

        response.addCookie(cookie);
    }
}
```

# 3. cookie 的细节 - 删除 cookie

## Tip：Cookie细节

- 一个Cookie只能标识一种信息，它至少含有一个标识该信息的名称（NAME）和设置值（VALUE）。
- 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。
- 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。
- 如果创建了一个cookie，并将他发送到浏览器，默认情况下它是一个会话级别的cookie（即存储在浏览器的内存中），用户退出浏览器之后即被删除。若希望浏览器将该cookie存储在磁盘上，则需要使用maxAge，并给出一个以秒为单位的时间。将最大时效设为0则是命令浏览器删除该cookie。
- 注意，删除cookie时，path必须一致，否则不会删除

本案例涉及2个文件。故事是记录并显示上次访问时间，点击清除超链接，清除上次访问时间，如下：

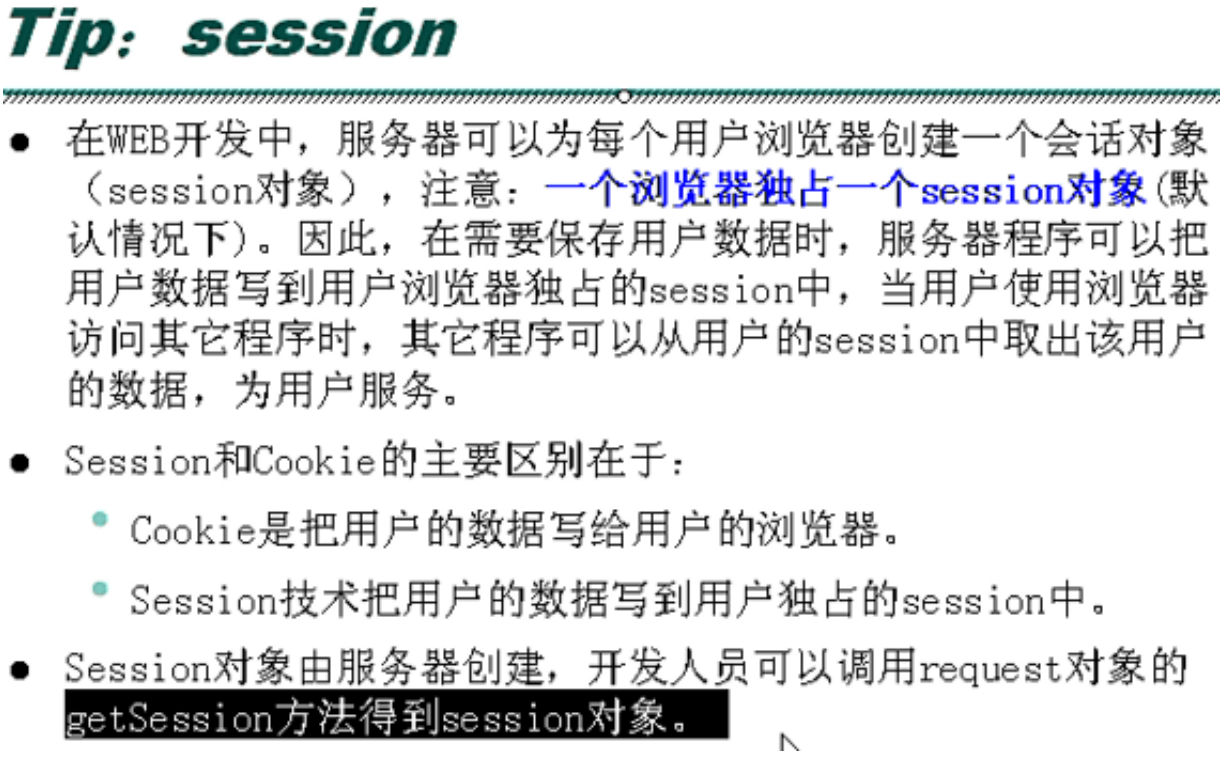
注意：设置相同的 **cookie** 的名字和 **path**，才能删除 **cookie**，可参考`CookieDemo2`的代码。

```java
// 代表网站首页
public class CookieDemo1 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();
        out.print("<a href='/day07/servlet/CookieDemo2'>清除上次访问时间</a
><br/>");
        out.print("您上次访问的时间是: ");

        // 获取用户的时间 cookie
        Cookie[] cookies = request.getCookies();
        for (int i = 0; cookies != null && i < cookies.length; i++) {
            if (cookies[i].getName().equals("lassAccessTime")) {
                long cookieValue = Long.parseLong(cookies[i].getValue
()); // ms

   // 得到用户的上次访问时间
                Date date = new Date(cookieValue);
                out.print(date.toLocaleString()); // 转换成本地习惯的日期
            }
        }

        // 给用户回送最新的访问时间
        Cookie cookie = new Cookie("lassAccessTime", System.currentTimeMi
llis()
                + "");
        cookie.setMaxAge(1 * 30 * 24 * 3600); // 设置有效期 秒 -- 1个月
        cookie.setPath("/day07"); // 设置该cookie适用范围

        response.addCookie(cookie);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

```
public class CookieDemo2 extends HttpServlet {

    // 清除 cookie
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        Cookie cookie = new Cookie("lassAccessTime", "");
        cookie.setMaxAge(0);
        cookie.setPath("/day07");

        response.addCookie(cookie);

        response.sendRedirect("/day07/servlet/CookieDemo1");
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {

        doGet(request, response);
    }
}
```

浏览器输入 `http://localhost:8080/day07/servlet/CookieDemo1` 查看结果。

# 4. Session 对象



**Cookie** 和 **Session** 只为**1**次会话**(**简单的说：打开浏览器访问网站**--**关闭访问那个网站的浏览器，这叫会话**)**而服务。可从下面的例子验证。

案例： 购买和结账，涉及3个文件如下：

首页：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'index.jsp' starting page</title>
  </head>

  <body>
    <a href="/day07/servlet/SessionDemo1">购买</a>
    <a href="/day07/servlet/SessionDemo2">结账</a>
  </body>
</html>
```

购买处理：

```
// 购买
public class SessionDemo1 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        HttpSession session = request.getSession(); // 创建或获取session
        session.setAttribute("name", "洗衣机");
    }
}
```

结账处理：

```
// 结账
public class SessionDemo2 extends HttpServlet {
 public void doGet(HttpServletRequest request, HttpServletResponse respon
se)
     throws ServletException, IOException {

   // 以后用 filter 解决网站乱码问题
   response.setCharacterEncoding("UTF-8");
   response.setContentType("text/html;charset=UTF-8");
   PrintWriter out = response.getWriter();

   HttpSession session = request.getSession(false); //只获取session
   String product = (String) session.getAttribute("name");

   out.write("您购买的商品是: " + product);
 }
}
```

浏览器中输入`http://localhost:8080/day07/index.jsp`查看结果。
上述例子并未达到方立勋老师所说的效果。我开启2个浏览器页面，一个点购买，另一个浏览器点结账，仍可收到session信息。
但若开启1个谷歌，1个IE，则试验成功。

## 4.1 配置 session 存活时间

创建 session 的时机：客户访问到`getSession()`方法时， 服务器就创建 session。
session 销毁的时机： 30分钟没人用这个 session 了，session 将被销毁，即使会话不关。

2种方式：web.xml 配置方式 和 代码方式。

web.xml 文件加入如下配置(存活10分钟，默认30分钟)：

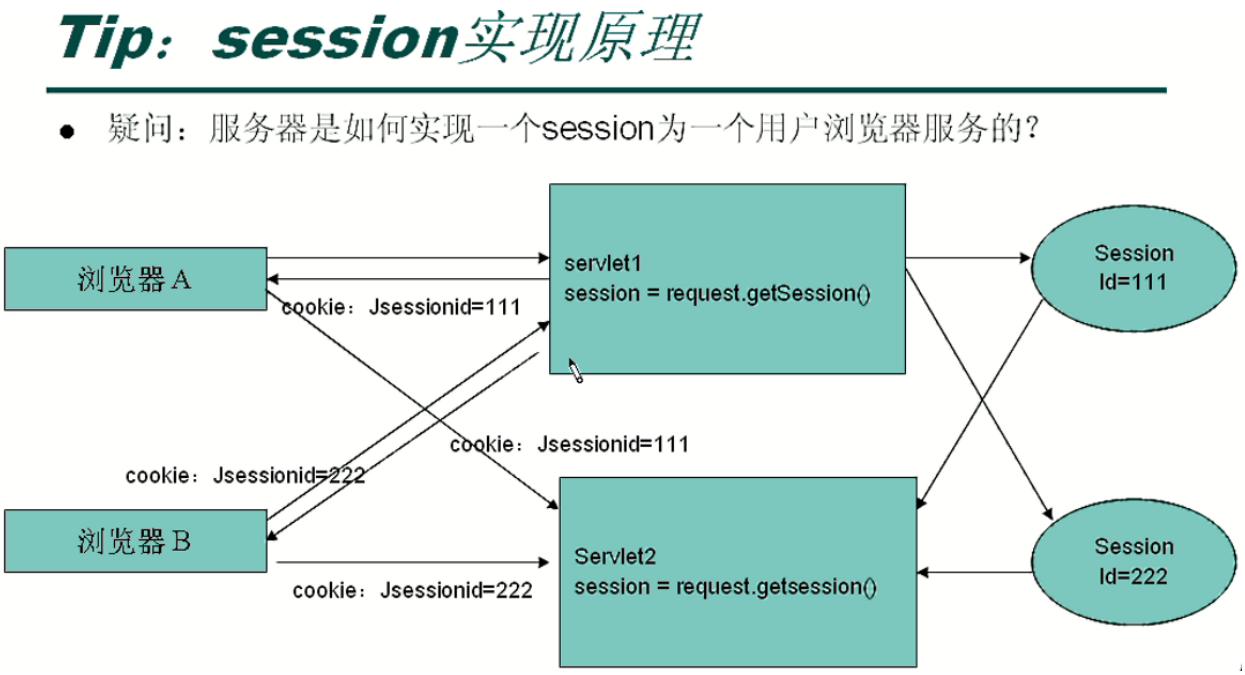
```
<session-config>
  <session-timeout>10</session-timeout>
</session-config>
```

代码方式：

摧毁session： 在 SessionDemo1 中加入 `session.invalidate();`
不创建，只获取session：正常的行为：“用户第一次访问***getSession()***就创建***session,***第***2***次访问时就不创建***session***而只获取***session***。”
但`request.getSession(false);`的执行会导致永远不创建session, 而只获取

session。

## 4.2 session 工作原理(基于 cookie，且无有效期，即浏览器关掉 cookie 就消失)




故事：想实现关掉浏览器(关会话)后，重新登录，也能看到所购买的商品。

原理分析：**getSession()**创建一个**session**时，还生成了一个**cookie**，这个**cookie**里包含了当前会话**session**的**id**,既`JSESSIONID`，并通过**response**写回给客户机浏览器，但没设定`setMaxAge()`，意味着关闭浏览器后这个**cookie**就消失了，所以重新登录后，浏览器就不会带着名为`JSESSIONID`的**cookie**找服务器，这样服务器就无法通过**id**号找到存放在服务器内的对应的**session**，故而无法找到客户原来买的东西了。

解决办法：重写名为`JSESSIONID`的**cookie**，设置其路径，设置其**MaxAge**，代码如下：

```java
// 购买
public class SessionDemo1 extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        HttpSession session = request.getSession();

        // 覆盖 getSession 时创建的 cookie(里面包含 id)
        String sessionid = session.getId();
        Cookie cookie = new Cookie("JSESSIONID", sessionid);
        cookie.setPath("/day07");
        cookie.setMaxAge(30 * 60);
        response.addCookie(cookie);

        session.setAttribute("name", "洗衣机");

        // session.invalidate(); // 摧毁 session
    }
}
```

结果就是：打开一个浏览器“购买”，关闭该浏览器，半小时内再开这个浏览器的话，仍然能“结账”，既上一个浏览器购买的商品还在。

# 5. 用户禁用了 cookie，session就不好使了，做电商网站的要解决这个问题，解决方案为：URL 重写

浏览器禁用cookie的操作：工具 -> Internet选项 -> 隐私 -> 高级 -> 替代自动 cookie 处理 -> 阻止第一方和第三方 cookie

本实验需将**localhost(**浏览器不阻止**localhost)**改成**127.0.0.1**才行， 如浏览器地址改为`http://127.0.0.1:8080/day07/index.jsp`。禁用 cookie 后，session 就不起作用了。

解决方案为：**URL** 重写
不同用户点击首页的超链接时，超链接里弄上 session 的 id 号。用户点超链接时，这个 session 的 id 号通过超链接传送给服务器。

重写 URL，新做一个 WelcomeSession 模拟首页，里面带了`response.encodeURL()`，它能自动把用户 session id 写入指定的 URL。下面代码能做到当用户禁用cookie时，通过超链接携带 session id 的功能。代码如下：

```
public class WelcomeSession extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        request.getSession();

        // 10个人访问相同的代码，但有10个不同的 session，超链接里就有10个不同
的 session id
        // 在 URL 后面加 session id
        String url1 = response.encodeURL("/day07/servlet/SessionDemo1");
        String url2 = response.encodeURL("/day07/servlet/SessionDemo2");

        out.print("<a href='" + url1 + "'>购买</a><br/>");
        out.print("<a href='" + url2 + "'>结账</a>");
    }
```

浏览器输入`http://localhost:8080/day07/servlet/WelcomeSession`点击购买，结账，结账时仍然能显示洗衣机。

# 6. session 细节：IE7 和 IE8以上浏览器在session上的不同表现

同一个session的情形：

- 开启浏览器，又开启一个新的选项卡窗口；
- 同一个浏览器，点开一个超链接窗口。

只有手动开启2个浏览器时(IE7)，会有不同的session。IE8以后的浏览器即使开2个浏览器，默认还是同一个session。我现在的机器就是这种情况(我的是IE11)。方老师为了讲课用的是IE7。

# 7. session - 简单的购物案例(一定要看看)

这个购物案例，我们在 cn.wk.shopping 包中建立 servlet 程序。
此案例涉及3个servlet程序：

- ListBookServlet 首页，列出所有书；

- BuyServlet 完成购买(购物车)；
- ListCartServlet 显示用户已购买的商品。

代码如下( 通过 **session** 一个劲滴传递购书信息，里面还涉及了用户禁用**cookie**后的处理，即 **URL** 重定向 )：

```java
// 代表网站首页，列出所有书
public class ListBookServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        request.getSession();

        out.print("本网站有如下商品：<br/>");
        Map<String, Book> map = Db.getAll();
        for (Map.Entry<String, Book> entry : map.entrySet()) {
            Book book = entry.getValue();

            String url = response.encodeURL("/day07/servlet/BuyServlet?id
="
                    + book.getId());

            // 正常情况下，输出时，这里通常要跳到 jsp，然后美化，本案例先不这样
做了
            // target='_blank' 打开新窗口
            out.print(book.getName() + "<a href='" + url
                    + "' target='_blank'>购买</a><br/>");
        }
    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }
}

// 模拟数据库
class Db {
    private static Map<String, Book> map = new LinkedHashMap<String, Book
>();
    static {
        map.put("1", new Book("1", "JavaWeb 开发", "老张", 39));
        map.put("2", new Book("2", "JDBC 开发", "老黎", 39));
        map.put("3", new Book("3", "Spring 开发", "老张", 39));
        map.put("4", new Book("4", "Hibernate 开发", "老方", 39));
        map.put("5", new Book("5", "Ajax 开发", "老毕", 39));
    }
```

```
    public static Map<String, Book> getAll() {
        return map;
    }
}

// bean
class Book implements Serializable {
    private String id;
    private String name;
    private String author;
    private double price;

    public Book() {
        super();
    }

    public Book(String id, String name, String author, double price) {
        super();
        this.id = id;
        this.name = name;
        this.author = author;
        this.price = price;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public double getPrice() {
        return price;
    }
```

```
    public void setPrice(double price) {
        this.price = price;
    }
}
```

```java
// 完成购买
public class BuyServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        String id = request.getParameter("id");
        Book book = Db.getAll().get(id);

        // 保存用户买的书，一会“结账”时用
        HttpSession session = request.getSession(false);

        /*
         * 1. 手工以cookie形式发sessionid，以解决关闭浏览器后，上次买的东西还
在(未解决)
         *
         * 2. 若用户关闭cookie，要用 URL
         * 重写方式(需在ListBookServlet首页中就得创建session)，重写所有超链接
(方老师已解决)
         */

        // 从session得到用户用于保存所有书的集合(购物车)
        List list = (List) session.getAttribute("list");
        if (list == null) {
            list = new ArrayList();
            session.setAttribute("list", list);
        }
        list.add(book);

        /*
         * request.getRequestDispatcher("/servlet/ListCartServlet").forwa
rd(
         * request, response); // 不能转发，浏览器刷新时会把上次干的事重新干
一次
         */

        // 用重定向跳到购物车
        // request.getContextPath() = /day07

        String url = response.encodeRedirectURL(request.getContextPath()
                + "/servlet/ListCartServlet");
        response.sendRedirect(url);
    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
```

```
    }
}
```

```
// 显示用户已购买的商品
public class ListCartServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        // 已购买信息在session的list中
        HttpSession session = request.getSession(false); // 不创建sessio
n 只获取
        if (session == null) {
            out.write("您没购买任何商品！！");
            return;
        }

        out.write("您购买了如下商品：<br/>");
        List<Book> list = (List) session.getAttribute("list");
        for (Book book : list) {
            out.write(book.getName()+"<br/>");
        }
    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

浏览器输入首页地址`http://127.0.0.1:8080/day07/servlet/ListBookServlet`查看结果，里面还进行了用户禁用cookie的处理，读者可以自己禁用自己浏览器的**cookie**功能试验一下，记得试验完后改回来！

# 8. 利用 session 完成用户登陆

故事：用户登录后，登录成功的这种状态，其他 servlet 程序也得知道，故需 session 技术。

此案例涉及5个文件：

- index.jsp 首页
- login.html 登录页
- cn.wk.login.User bean
- cn.wk.login.LoginServlet 登录处理
- cn.wk.login.LogoutServlet 注销处理

index.jsp 首页:

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'index.jsp' starting page</title>
  </head>

  <body>
    欢迎您：${user.username}<br/>
    <a href="/day07/login.html">登录</a>
    <a href="/day07/servlet/LogoutServlet">退出登录</a>

    <br/><br/><br/>
  </body>
</html>
```

login.html 登录页:

```
<!DOCTYPE html>
<html>
  <head>
    <title>login.html</title>

    <meta name="keywords" content="keyword1,keyword2,keyword3">
    <meta name="description" content="this is my page">
    <meta name="content-type" content="text/html; charset=UTF-8">
  </head>

  <body>
    <form action="/day07/servlet/LoginServlet" method="post">
        用户名：<input type="text" name="username"><br/>
        密码：<input type="text" name="password"><br/>
        <input type="submit" value="登录">
    </form>
  </body>
</html>
```

cn.wk.login.User bean：

```java
// bean
public class User {
    private String username;
    private String password;

    public User() {
        super();
    }

    public User(String username, String password) {
        super();
        this.username = username;
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

cn.wk.login.LoginServlet 登录处理:

```java
// 登录处理
public class LoginServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html;charset=UTF-8");
        PrintWriter out = resp.getWriter();

        // 接收用户名和密码
        String username = req.getParameter("username");
        String password = req.getParameter("password");

        // 检查数据库有没有
        List<User> list = DB.getAll();
        for (User user : list) {

            if (user.getUsername().equals(username)
                    && user.getPassword().equals(password)) {
                // 若登录成功,则向session存一个登录标记
                req.getSession().setAttribute("user", user);
                resp.sendRedirect("/day07/index.jsp");
                return;
            }
        }

        out.write("用户名和密码不对！！");
        out.write("<a href='/day07/login.html'>返回</a>");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}

// 模拟数据库
class DB {
    public static List<User> list = new ArrayList<User>();
    static {
        list.add(new User("aaa", "123"));
        list.add(new User("bbb", "123"));
        list.add(new User("ccc", "123"));
    }

    public static List<User> getAll() {
```

```
        return list;
    }
}
```

cn.wk.login.LogoutServlet 注销处理:

```
// 用户注销
public class LogoutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {

        HttpSession session = req.getSession();
        if(session==null){
            resp.sendRedirect("/day07/index.jsp");
            return;
        }

        // 移除 session 中的 user
        session.removeAttribute("user");
        resp.sendRedirect("/day07/index.jsp");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse res
p)
            throws ServletException, IOException {
        doGet(req, resp);
    }
}
```

浏览器输入 `http://localhost:8080/day07/index.jsp` 查看结果。

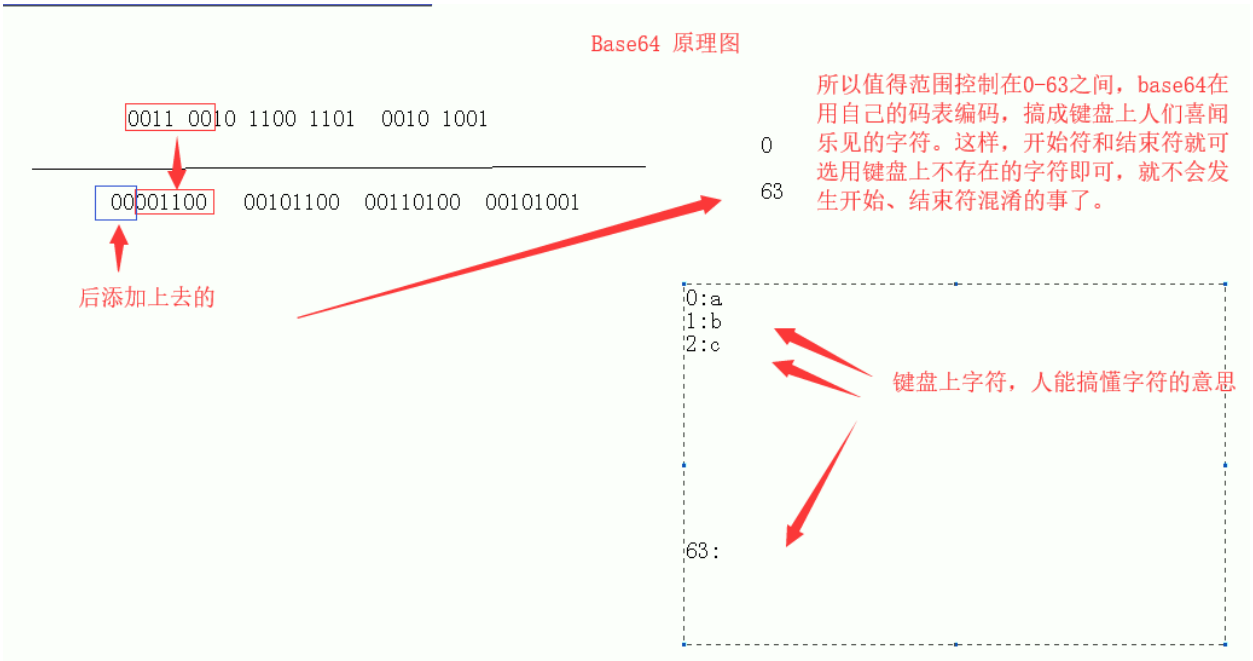
# 9. 客户端防表单重复提交和服务器端 session 防表单重复提交 - 涉及 Base64Encoder，令牌发生器，单例设计模式

这里只叙述服务器端 session 防表单重复提交。

思路：由服务器**servlet**程序生成表单，打给客户浏览器。服务器的**servlet**将生成一个随机数，并记录下这个随机数，并把这个随机数随表单一同发给客户机浏览器。客户机浏览器提交这个表单时，随同那个随机数一起提交。服务器接收到随机数后，与服务器自己所存的随机数进行对比，若能匹配，则接受客户机提交的

表单，同时在服务器端删除那个随机数；否则，若服务器端的随机数与客户机发来的不匹配，或者服务器端根本就没那个客户机发来的随机数，则服务器不接受提交来的表单，并返回通知：“您已重复提交，服务器不接受！”。

涉及到 **Base64** 编码 ，文件上传下载，设置数据开始和结束标志，都必须要对传输数据进行 Base64 编码，再加上开始结束标志，再传。若无这个编码，所设置的开始结束标志可能在所传输的数据中出现，会出问题。故而 Base64 编码非常重要，原理如下图：




又涉及到 解决**Eclipse**中无法直接使用**Base64Encoder**的问题，请参考 https://blog.csdn.net/u011514810/article/details/72725398

该案例涉及3个文件：

- cn.wk.form.FormServlet 产生表单；
- WebRoot/form.jsp 表单页；
- cn.wk.form.DoFormServlet 处理表单提交请求，判断是否重复提交。

FormServlet.java 文件中还涉及了 Base64Encoder，令牌发生器，单例设计模式。以上文件全部代码如下：

```java
// 产生表单
public class FormServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        // 产生随机数(表单号)
        TokenProcessor tp = TokenProcessor.getInstance();
        String token = tp.generateToken();

        request.getSession().setAttribute("token", token);

        request.getRequestDispatcher("/form.jsp").forward(request, respon
se);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }
}

// 随机数发生器
class TokenProcessor { // 令牌 单例设计模式：随机数只找一个令牌发生器对象产生
    /* 单例设计模式步骤 */

    // 1. 把构造方法私有
    private TokenProcessor() {
    }

    // 2. 自己创建1个
    private static final TokenProcessor instance = new TokenProcessor();

    // 3. 对外暴露1个方法，允许获取上面创建的对象
    public static TokenProcessor getInstance() {
        return instance;
    }

    public String generateToken() {

        // 因为获得的随机数长度不同，想获得相同长度的随机数可通过 md5 获字符串
的摘要(指纹)来实现
        // 指纹长度是固定的，共 128 bit
        String token = System.currentTimeMillis() + new Random().nextInt
() + "";
        try {
            MessageDigest md = MessageDigest.getInstance("md5");
            byte[] md5 = md.digest(token.getBytes());
```

```
            // return md5 + "" 这是不行的，它是要查码表的。

            // 应该用 base64 编码：任何数据通过该编码，都返回一个明文字符串
            BASE64Encoder encoder = new BASE64Encoder();
            return encoder.encode(md5);

        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
}
```

WebRoot/form.jsp 表单页：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>My JSP 'form.jsp' starting page</title>
  </head>

  <body>
    <form action="/day07/servlet/DoFormServlet" method="post">
        <input type="hidden" name="token" value="${token}"> <!-- 获取session中的token -->
        用户名：<input type="text" name="username"><br/>
        <input type="submit" value="提交">
    </form>
  </body>
</html>
```

```java
// 处理表单提交请求，判断是否重复提交
public class DoFormServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        boolean b = isTokenValid(request);
        if (!b) {
            System.out.println("请不要重复提交");
            return;
        }

        // 有效提交的话，就处理提交并在服务器session里删除表单号
        request.getSession().removeAttribute("token");
        System.out.println("向数据库中注册用户~~~~");
    }

    // 判断表单号是否有效
    private boolean isTokenValid(HttpServletRequest request) {
        String client_token = request.getParameter("token"); // 客户机通过
表单带来的token
        if (client_token == null) {
            return false;
        }
        String server_token = (String) request.getSession().getAttribute(
                "token");
        if (server_token == null) {
            return false;
        }

        if (!client_token.equals(server_token)) {
            return false;
        }

        // 闯过3关
        return true;
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }
}
```

浏览器输入`http://localhost:8080/day07/servlet/FormServlet`查看效果。刷新，后退，我们的 servlet 均能阻止用户重复提交表单。

# 10. 利用session校验图片认证码

登录页面 login2.html

```
<!DOCTYPE html>
<html>
<head>
<title>register.html</title>
<meta http-equiv="content-type" content="text/html;charset=UTF-8">
<script type="text/javascript">
    function changeImage(img) {
        img.src = img.src + "?" + new Date().getTime();
    }
</script>
</head>

<body>
    <form action="/day07/servlet/RegisterServlet" method="post">
        用户名：<input type="text" name="username"><br />
        密码：<input type="password" name="password"><br />
        认证码：<input type="text"  name="checkcode">

        <img src="/day07/servlet/ImageServlet" onclick="changeImage(thi
s)"
        alt="换个图片" style="cursor:hand"><br />
            <input type="submit" value="注册">
    </form>
</body>
</html>
```

产生随机图片的 ImageServlet:

```java
public class ImageServlet extends HttpServlet {
    // ctrl + shift + x 小写变大写
    public static final int WIDTH = 120;
    public static final int HEIGHT = 35;

    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        BufferedImage image = new BufferedImage(WIDTH, HEIGHT,
                BufferedImage.TYPE_INT_RGB);

        Graphics g = image.getGraphics();

        // 1. 设置背景色
        setBackGround(g);

        // 2. 设置边框
        setBorder(g);

        // 3. 画干扰线
        drawRandomLine(g);

        // 4. 写随机数
        String checkcode = drawRandomNum((Graphics2D) g);
        // --------> 已改动：产生图片时，向session写入随机数
        request.getSession().setAttribute("checkcode", checkcode);

        // 5. 图形写给浏览器
        response.setContentType("image/jpeg");
        // 发头控制浏览器不要缓存
        response.setDateHeader("expries", -1);
        response.setHeader("Cache-Control", "no-cache");
        response.setHeader("Pragma", "no-cache");

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

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {
        doGet(request, response);
    }

    private String drawRandomNum(Graphics2D g) {
        g.setColor(Color.RED);
        g.setFont(new Font("宋体", Font.BOLD, 20));

        // unicode 汉字码表范围 [\u4e00-\u9fa5]
        // 常用汉字
```

```
        String base = "\u4e00\u4e86\u662f\u6211\u4e0d\u5728\u4eba\u4eec\u
6709\u6765\u4ed6\u8fd9\u4e0a\u7740\u4e2a\u5730\u5230\u5927\u91cc\u8bf4\u5
c31\u53bb\u5b50\u5f97\u4e5f\u548c\u90a3\u8981\u4e0b\u770b\u5929\u65f6\u8f
c7\u51fa\u5c0f\u4e48\u8d77\u4f60\u90fd\u628a\u597d\u8fd8\u591a\u6ca1\u4e3
a\u53c8\u53ef\u5bb6\u5b66\u53ea\u4ee5\u4e3b\u4f1a\u6837\u5e74\u60f3\u751f
\u540c\u8001\u4e2d\u5341\u4ece\u81ea\u9762\u524d\u5934\u9053\u5b83\u540e
\u7136\u8d70\u5f88\u50cf\u89c1\u4e24\u7528\u5979\u56fd\u52a8\u8fdb\u6210
\u56de\u4ec0\u8fb9\u4f5c\u5bf9\u5f00\u800c\u5df1\u4e9b\u73b0\u5c71\u6c11
\u5019\u7ecf\u53d1\u5de5\u5411\u4e8b\u547d\u7ed9\u957f\u6c34\u51e0\u4e49
\u4e09\u58f0\u4e8e\u9ad8\u624b\u77e5\u7406\u773c\u5fd7\u70b9\u5fc3\u6218
\u4e8c\u95ee\u4f46\u8eab\u65b9\u5b9e\u5403\u505a\u53eb\u5f53\u4f4f\u542c
\u9769\u6253\u5462\u771f\u5168\u624d\u56db\u5df2\u6240\u654c\u4e4b\u6700
\u5149\u4ea7\u60c5\u8def\u5206\u603b\u6761\u767d\u8bdd\u4e1c\u5e2d\u6b21
\u4eb2\u5982\u88ab\u82b1\u53e3\u653e\u513f\u5e38\u6c14\u4e94\u7b2c\u4f7f
\u5199\u519b\u5427\u6587\u8fd0\u518d\u679c\u600e\u5b9a\u8bb8\u5feb\u660e
\u884c\u56e0\u522b\u98de\u5916\u6811\u7269\u6d3b\u90e8\u95e8\u65e0\u5f80
\u8239\u671b\u65b0\u5e26\u961f\u5148\u529b\u5b8c\u5374\u7ad9\u4ee3\u5458
\u673a\u66f4\u4e5d\u60a8\u6bcf\u98ce\u7ea7\u8ddf\u7b11\u554a\u5b69\u4e07
\u5c11\u76f4\u610f\u591c\u6bd4\u9636\u8fde\u8f66\u91cd\u4fbf\u6597\u9a6c
\u54ea\u5316\u592a\u6307\u53d8\u793e\u4f3c\u58eb\u8005\u5e72\u77f3\u6ee1
\u65e5\u51b3\u767e\u539f\u62ff\u7fa4\u7a76\u5404\u516d\u672c\u601d\u89e3
\u7acb\u6cb3\u6751\u516b\u96be\u65e9\u8bba\u5417";

        StringBuffer sb = new StringBuffer();

        int x = 5; // 从这开始写
        for (int i = 0; i < 4; i++) {
            String ch = base.charAt(new Random().nextInt(base.length
())) + "";
            sb.append(ch);

            // -30 -- 30度旋转设定
            int degree = new Random().nextInt() % 30;
            g.rotate(degree * Math.PI / 180, x, 20); // 设置旋转弧度
            g.drawString(ch, x, 20);
            g.rotate(-degree * Math.PI / 180, x, 20);
            x += 30;
        }
        return sb.toString();
    }

    private void drawRandomLine(Graphics g) {
        g.setColor(Color.GREEN);
        // 4-5条干扰线
        for (int i = 0; i < 5; i++) {

            int x1 = new Random().nextInt(WIDTH);
            int y1 = new Random().nextInt(HEIGHT);

            int x2 = new Random().nextInt(WIDTH);
            int y2 = new Random().nextInt(HEIGHT);
```

```
            g.drawLine(x1, y1, x2, y2);
        }
    }

    private void setBorder(Graphics g) {
        g.setColor(Color.BLUE);
        g.drawRect(1, 1, WIDTH - 2, HEIGHT - 2);
    }

    private void setBackGround(Graphics g) {
        g.setColor(Color.WHITE);
        g.fillRect(0, 0, WIDTH, HEIGHT);
    }
}
```

RegisterServlet：处理注册请求之前，校验认证码是否有效

```
public class RegisterServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse res
ponse)
            throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");

        // 处理注册请求之前，校验认证码是否有效
        String client_checkcode = request.getParameter("checkcode");
        String server_checkcode = (String) request.getSession().getAttrib
ute(
                "checkcode");

        if (client_checkcode == null) {
            System.out.println("客户机无checkcode");
            return;
        }
        if (server_checkcode == null) {
            System.out.println("服务器无checkcode");
            return;
        }
        if (client_checkcode.equals(server_checkcode)) {
            System.out.println("处理客户机请求！");
        } else {
            System.out.println("验证码不匹配！");
        }
    }

    public void doPost(HttpServletRequest request, HttpServletResponse re
sponse)
            throws ServletException, IOException {

        doGet(request, response);
    }
}
```

# 11. 三个域对象的总结 - request, session 和 servletContext 这三个域，分别应该啥时候用？

request, session 和 servletContext 这三个域，分别应该啥时候用？

- request 存储域：一个程序如 html 或 jsp 产生了数据，给我们的 servlet 程序后显示完了，没用了，这时用request容器；
- session 存储域：数据产生了，给用户看后，等一会还要用，这时应该用

session 容器；
- servletContext(web 应用程序全局性的东西) 存储域：数据显示完了，除了一会还要用，还要给别人用(如：聊天室)，这时应该用 servletContext 容器。

方立勋老师又提到了 md5 算法。它不是加密算法，而是提取数据指纹的算法。应用场景：

- 用户密码通过 md5 得到结果后存入数据库，防止管理员窥探密码(说实话，我有点不信。为什么不用真正的加密算法加密密码，然后再用md5提取指纹呢(128bit，保证了长度都一样)？);
- 数据传输时使用 md5，这个在第9节防止重复提交表单中的令牌产生器类 `TokenProcessor` 代码里讲过。