第3章

Session与Cookie

Session与Cookie

本讲内容：

3.1 会话（Session）管理

3.2 Cookie及其应用

3.1 会话管理

3.1.1 理解状态与会话

3.1.2 会话管理机制

3.1.3 常用HttpSession API

3.1.4 使用HttpSession对象

3.1.5 会话超时与失效

3.1.1 理解状态与会话

- 协议记住用户及其请求的能力称为状态（state）。
- 协议分成两种类型：
  - 有状态的
  - 无状态的

3.1.1 理解状态与会话

1. HTTP协议是一种无状态的协议

- HTTP服务器对用户的每个请求和响应都是作为一个分离的事务处理。

- 服务器无法确定多个请求是来自相同的用户还是不同的用户。这意味着服务器不能在多个请求中维护用户的状态。

3.1.1 理解状态与会话

2.购物车应用

在某些情况下服务器不需要记住用户，HTTP无状态的特性对这样的应用会工作得很好。但在一些特定的Web应用程序中，用户与服务器的交互就需要有状态。

——典型的例子是购物车应用：

一个用户可以多次向购物车中添加商品，也可以清除商品。在处理过程中，服务器应该能够显示购物车中商品并计算总价格。为了实现这一点，服务器必须跟踪所有的请求并把它们与用户关联。

3.1.1 理解状态与会话

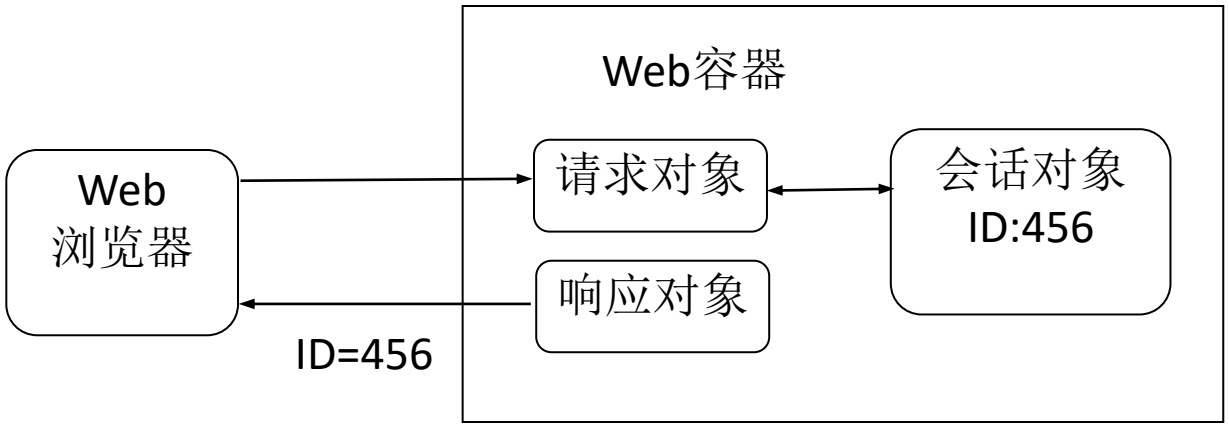
- 会话（session）是用户与服务器之间的不间断的请求响应序列。

当一个用户向服务器发送第一个请求时就开始了一个会话。对该用户之后的每个请求，服务器能够识别出请求来自于同一个用户。

当用户明确结束会话或服务器在一个预定义的时限内没从用户接收任何请求时，会话就结束了。当会话结束后，服务器就删除了用户以及用户的请求。

3.1.2 会话管理机制

Web 容器通过 javax.servlet.http.HttpSession 接口抽象会话的概念。该接口由容器实现并提供了一个简单的管理用户会话的方法。使用HttpSession对象管理会话的过程如图所示。



3.1.2 会话管理机制

（1）当用户向服务器发送第一个请求时，Web容器就可以使用请求对象为该用户创建一个HttpSession会话对象，并将请求对象与该会话对象关联。

■服务器在创建会话对象时为其指定一个唯一标识符，称为会话ID，它是一个32位的十六进制数，作为该客户的唯一标识。

■如何确定会话处于新建状态？可以使用HttpSession的isNew()方法来确定。

3.1.2 会话管理机制

（2）当服务器向用户发送响应时，服务器将该会话ID与响应数据一起发送给客户，这是通过Set-Cookie响应头实现的，响应消息可能为：

```
HTTP/1.1 200 OK
Set-Cookie:JSESSIONID=61C4F23524521390E70993E5120263C6
Content-Type:text/html
…
```

■这里，JSESSIONID的值即为会话ID，它是32位的十六进制数。

3.1.2 会话管理机制

（3）用户在接收到响应后将会话ID存储在浏览器的内存中。当用户再次向服务器发送一个请求时，它将通过Cookie请求头把会话ID与请求一起发送给服务器。这时请求消息可能为：

```
POST /helloweb/selectProduct.do HTTP/1.1
Host:www.mydomain.com
Cookie: JSESSIONID=61C4F23524521390E70993E5120263C6
...
```

3.1.2 会话管理机制

（4）服务器接收到请求后，从请求对象中取出会话ID，在服务器中查找之前创建的会话对象，找到后将该请求与之前创建的ID值相同的会话对象关联起来。

上述过程的第（2）到第（4）步一直保持重复。

■如果用户在指定时间没有发送任何请求，服务器将使会话对象失效。

一旦会话对象失效，即使客户再发送同一个会话ID，会话对象也不能恢复。对于服务器来说，此时客户的请求被认为是第一次请求（如第1步），它不与某个存在的会话对象关联。

解决：服务器重新为该客户创建一个新的会话对象。

3.1.2 会话管理机制

- 通过会话机制可以实现购物车应用。
- 当用户登录购物网站时，Web容器就为用户创建一个HttpSession对象。实现购物车的Servlet使用该会话对象存储用户的购物车对象，购物车中存储着用户购买的商品列表。
- 当用户向购物车中添加商品或删除商品时，Servlet就更新该列表。当用户要结账时，Servlet就从会话中检索购物车对象，从购物车中检索商品列表并计算总价格。一旦客户结算完成，容器就会关闭会话。

3.1.2 会话管理机制

通常用户发送一个请求，就会创建一个新的会话。因此，有多少个会话，服务器就会创建多少个**HttpSession**对象。

换句话说，对每个会话（用户）都有一个对应的**HttpSession**对象。然而，无需担心**HttpSession**对象与用户的关联，容器会为我们做这一点，一旦接收到请求，它会自动返回合适的会话对象。

注意：不能使用用户的IP地址唯一标识用户。因为，用户可能是通过局域网访问Internet。尽管在局域网中每个用户有一个IP地址，但对于服务器来说，用户的实际IP地址是路由器的IP地址，所以该局域网的所有用户的IP地址都相同！因此也就无法唯一标识用户。

3.1.3 常用 HttpSession API

- 下面是**HttpSession**接口中定义的常用方法。

```
public String getId()
public long getCreationTime()
public long getLastAccessedTime()
public boolean isNew()
public ServletContext getServletContext()
```

3.1.3 常用 HttpSession API

public void setAttribute (String name, Object value)

public Object getAttribute(String name)

public Enumeration getAttributeNames()

public void removeAttribute(String name)

public void setMaxInactiveInterval(int interval)

public int getMaxInactiveInterval()

public void invalidate()

3.1.4 使用 HttpSession 对象

- 使用会话对象通常需要三步：

1）创建或返回与客户请求关联的HttpSession对象。

2）在会话对象中添加或删除“名/值”对属性。

3）如果需要可使会话失效。

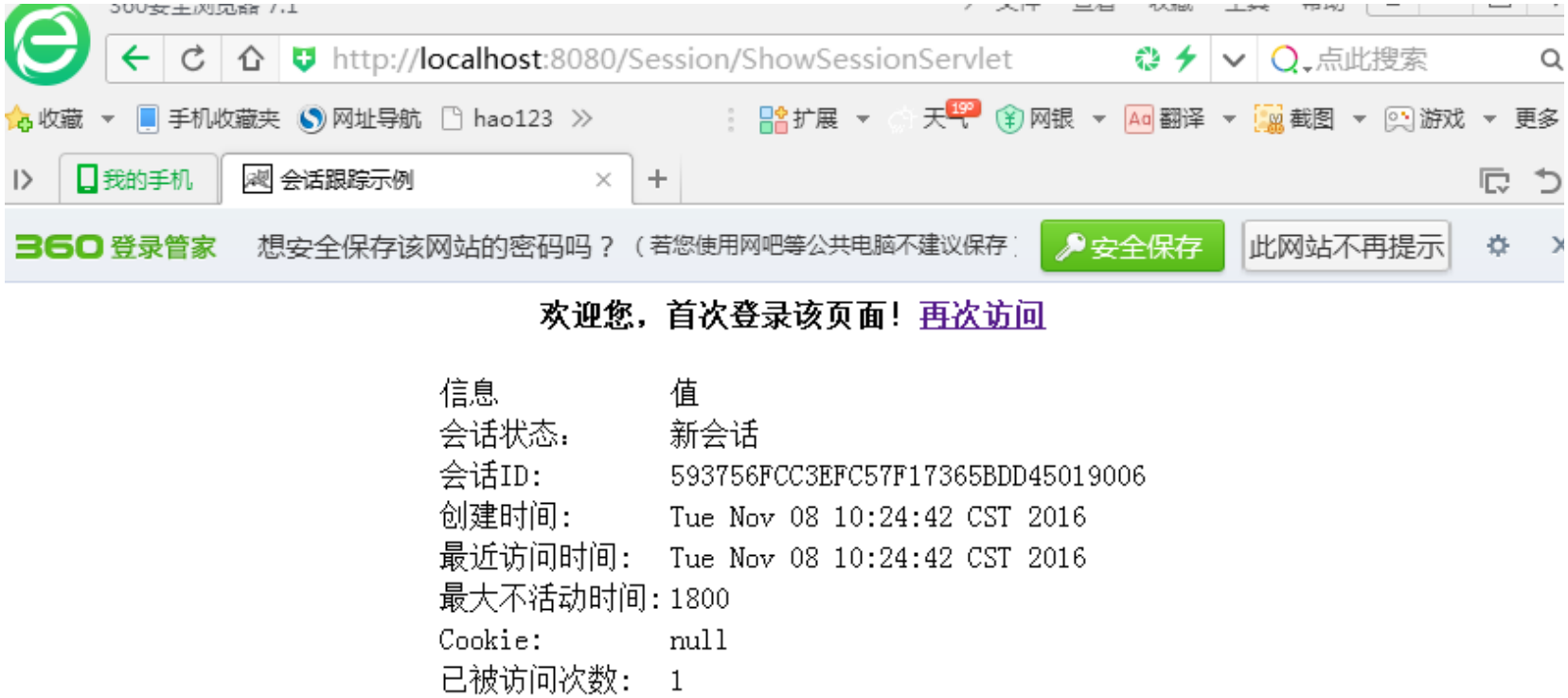
3.1.4 使用 HttpSession 对象

- 创建或返回HttpSession对象需要使用HttpServletRequest接口提供的getSession()方法，该方法有两种格式。

```
public HttpSession getSession(boolean create)
public HttpSession getSession()
```

3.1.4 使用 HttpSession 对象

■例3-1 通过Servlet完成显示客户会话的基本信息。



3.1.4 使用 HttpSession 对象

【例3-1】ShowSessionServlet.java程序。

■程序调用request.getSession()方法获取现存的会话，在没有会话的情况下创建新的会话。

■然后，在会话对象上查找类型为Integer的accessCount属性。如果找不到这个属性，则使用1作为访问计数。然后，对这个值进行递增，并用setAttribute()方法与会话关联起来。

ShowSessionServlet.java程序

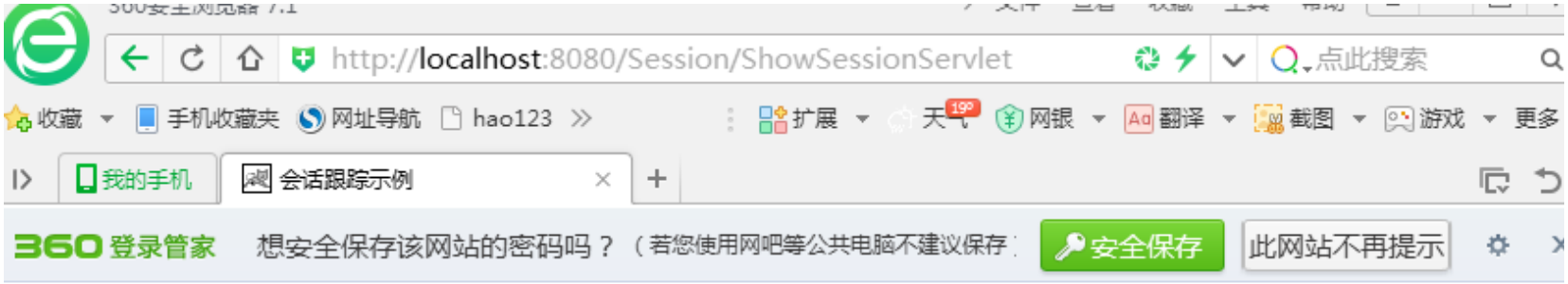
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;

public class ShowSessionServlet extends HttpServlet{
  public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    // 创建或返回用户会话对象
    HttpSession session = request.getSession(true);
    String heading = null;
    // 从会话对象中检索accessCount属性
    Integer accessCount = (Integer)session.getAttribute("accessCount");
    if(accessCount==null){
        accessCount = new Integer(1);
      heading = "欢迎您，首次登录该页面！";
    }else{
      heading = "欢迎您，再次访问该页面！";
      accessCount = accessCount+1;
    }
```

ShowSessionServlet.java程序

```
// 将accessCount作为属性存储到会话对象中
    session.setAttribute("accessCount",accessCount);
    PrintWriter out = response.getWriter();
    out.println("<html><head>");
    out.println("<title>会话跟踪示例</title></head>");
    out.println("<body><center>");
    out.println("<h4>"+heading
          +"<a href='ShowSessionServlet'>再次访问</a>"+"</h4>");
    out.println("<table border='0'>");
    out.println("<tr><td>信息</td><td>值</td>\n");
    String state = session.isNew()?"新会话":"旧会话";
    out.println("<tr><td>会话状态：<td>"+state+"\n");
    out.println("<tr><td>会话ID:<td>"+session.getId()+"\n");
    out.println("<tr><td>创建时间:<td>");
    out.println(""+new Date(session.getCreationTime())+"\n");
    out.println("<tr><td>最近访问时间:<td>");
    out.println(""+new Date(session.getLastAccessedTime())+"\n");
    out.println("<tr><td>最大不活动时间:<td>"+
    session.getMaxInactiveInterval()+"\n");
    out.println("<tr><td>Cookie:<td>"+request.getHeader("Cookie")+"\n");
    out.println("<tr><td>已被访问次数:<td>"+accessCount+"\n");
    out.println("</table>");
    out.println("</center></body></html>"); } }
```

3.1.4 使用 HttpSession 对象

■ 第一次访问该Servlet，显示如图所示的页面，此时计数变量accessCount值为1，Cookie请求头的值为null。

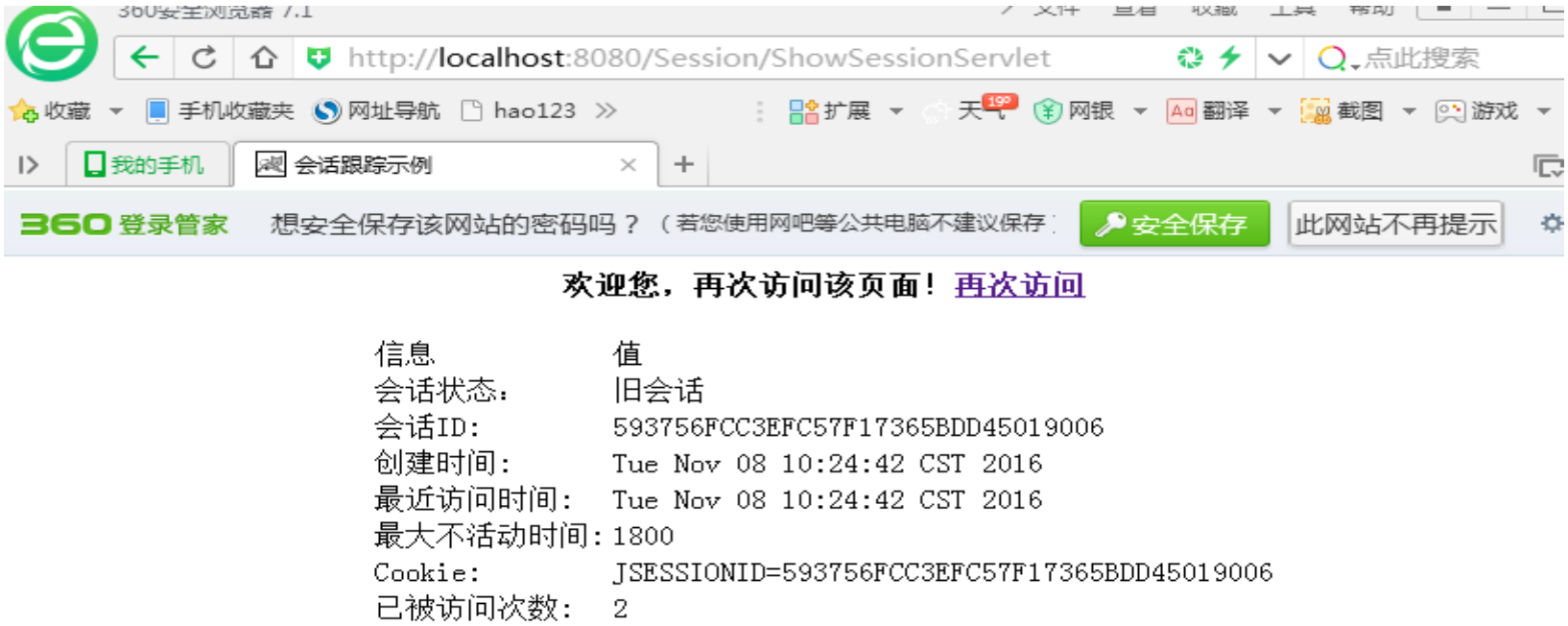


欢迎您，首次登录该页面！再次访问

信息	值
会话状态：	新会话
会话ID：	593756FCC3EFC57F17365BDD45019006
创建时间：	Tue Nov 08 10:24:42 CST 2016
最近访问时间：	Tue Nov 08 10:24:42 CST 2016
最大不活动时间：	1800
Cookie：	null
已被访问次数：	1

3.1.4 使用 HttpSession 对象

■再次访问页面（单击“再次访问”链接或刷新页面）计数变量accessCount值增1，但会话ID的值相同。



■注意：如果再打开一个浏览器窗口访问该Servlet，计数变量仍从1开始，因为等于开始了一个新的会话，服务器将为该会话创建一个新的会话对象并分配一个新的会话ID。

3.1.5 会话超时与失效

■会话对象会占用一定的系统资源，我们不希望会话在不需要的情况下长久保留。但是，HTTP协议没有提供任何机制让服务器知道用户已经离开，但可以规定当用户在一个指定的期限内处于不活动状态时，就将用户的会话终止，这称为会话超时。

3.1.5 会话超时与失效

■可以在DD文件中设置会话超时时间。

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

■**<session-timeout>**元素中指定以分钟为单位的超时期限。**0**或小于**0**的值表示会话永不过期。

■如果没有通过上述方法设置会话的超时期限，默认情况下是**30**分钟。如果用户在指定期间内没有执行任何动作，服务器就认为用户处于不活动状态并使会话对象无效。

3.1.5 会话超时与失效

- 在DD中设置的会话超时时间是针对Web应用程序中的所有会话对象有效，但有时可能需要对特定的会话对象指定超时时间，可使用**setMaxInactiveInterval()**方法。
- 注意该方法仅对调用它的会话有影响，其他会话的超时期限仍然是DD中设置的值。

3.1.5 会话超时与失效

- 例如，在购物车的应用中，我们希望在用户付款处理完成后结束会话。这样，当客户再次发送请求时，就会创建一个购物车中不包含商品的新的会话。
- 可使用**HttpSession**接口的**invalidate()**方法结束会话。

Session与Cookie

本讲内容：

3.1 会话（Session）管理

3.2 Cookie及其应用

3.2 Cookie及其应用

3.2.1 Cookie API

3.2.2 向客户端发送Cookie

3.2.3 从客户端读取Cookie

3.2 Cookie及其应用

- Cookie是用户访问Web服务器时服务器在用户硬盘上存放的信息。
- Cookie实际上是一小段文本信息，用户以后访问同一个Web服务器时，浏览器会把它们原样发送给服务器。
- 通过让服务器读取它原先保存到客户端的信息，网站能够为浏览者提供一系列的方便，例如，在线交易过程中标识用户身份、安全要求不高的场合避免用户登录时重复输入用户名和密码等。

3.2.1 Cookie API

- Cookie的管理需使用javax.servlet.http.Cookie类，构造方法如下：

  public Cookie(String name, String value)

- 参数name为Cookie名， value为Cookie的值，它们都是字符串。
- Cookie类的常用方法如下：

3.2.1 Cookie API

public String getName()

public String getValue()

public void setValue(String newValue)

public void setMaxAge(int expiry)

public int getMaxAge()

public void setDomain(String pattern)

public String getDomain()

3.2.2向客户端发送Cookie

■要把Cookie发送到客户端，先使用Cookie类的构造方法创建一个Cookie对象，通过setXxx()方法设置各种属性，通过响应对象的addCookie(cookie)把Cookie加入响应头。

■具体步骤如下：

1）创建Cookie对象。调用Cookie类的构造方法创建Cookie对象。下面语句创建了一个Cookie对象。

```
Cookie userCookie =
        new Cookie(“username”, “hacker”);
```

3.2.2向客户端发送Cookie

2）设置Cookie的最大存活时间。在默认情况下，发送到客户端的Cookie对象只是一个会话级别的Cookie，它存储在浏览器的内存中，用户关闭浏览器后Cookie对象将被删除。

■如果希望浏览器将Cookie对象存储到磁盘上，需要使用Cookie类的setMaxAge()方法设置Cookie的最大存活时间。

■下面的代码将userCookie对象的最大存活时间设置为一个星期。

```
userCookie.setMaxAge(60*60*24*7);
```

3.2.2向客户端发送Cookie

3）向客户发送Cookie对象。调用响应对象的addCookie()方法将Cookie添加到Set-Cookie响应头，如下所示。

```
response.addCookie(userCookie);
```

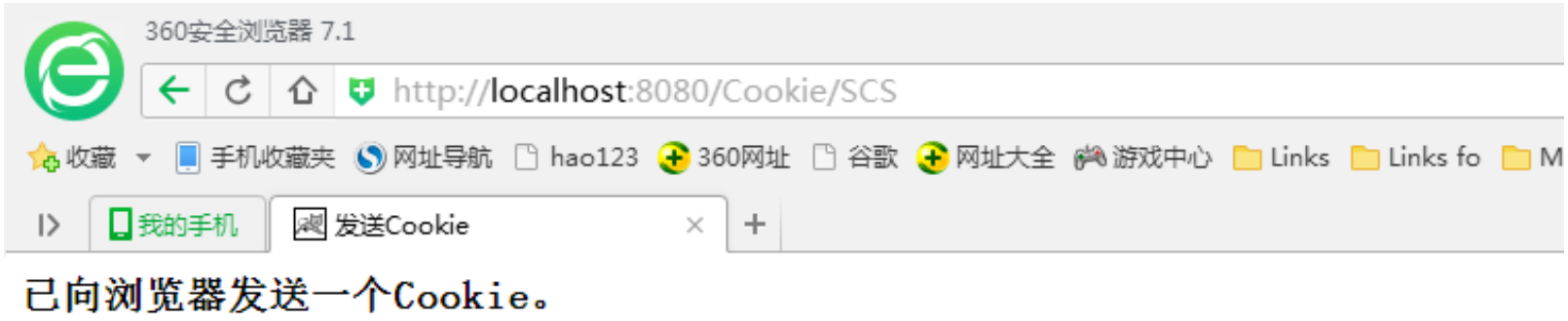
下面的例子是向客户发送一个Cookie对象。

【例3-2】SendCookieServlet.java程序。

SendCookieServlet.java

```
package com.demo;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SendCookieServlet extends HttpServlet{
  public void doGet(HttpServletRequest request,
HttpServletResponse response)
      throws IOException,ServletException{

      Cookie userCookie = new Cookie("username", "hacker");
      userCookie.setMaxAge(60*60*24*7);
      response.addCookie(userCookie);

 response.setContentType("text/html;charset=UTF-8");
 PrintWriter out = response.getWriter();
out.println("<html><title>发送Cookie</title>");
out.println("<body><h3>已向浏览器发送一个Cookie。
</h3></body>");
 out.println("</html>");
  }
}
```

运行效果


360安全浏览器 7.1
http://localhost:8080/Cookie/SCS
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
游戏中心
Links
Links fo
M
我的手机
发送Cookie
已向浏览器发送一个Cookie。

3.2.3 从客户端读取Cookie

■从客户端读入**Cookie**，调用请求对象的**getCookies()**方法，该方法返回一个**Cookie**对象的数组。

■大多数情况下，只需要用循环访问该数组的各个元素寻找指定名字的**Cookie**，然后对该**Cookie**调用**getValue()**方法取得与指定名字关联的值。

■具体步骤如下：

1）调用请求对象的**getCookies()**方法。该方法返回一个**Cookie**对象的数组。如果请求中不含**Cookie**，返回**null**值。

```
Cookie[] cookies=request.getCookies();
```

3.2.3 从客户端读取Cookie

2）对Cookie数组循环。有了Cookie对象数组后，就可以通过循环访问它的每个元素，然后调用每个Cookie的getName()方法，直到找到一个与希望的名称相同的对象为止。

■找到所需要的Cookie对象后，一般要调用它的getValue()方法，并根据得到的值做进一步处理。

下面的例子是从客户端读取Cookie。

【例3-3】ReadCookieServlet.java程序。

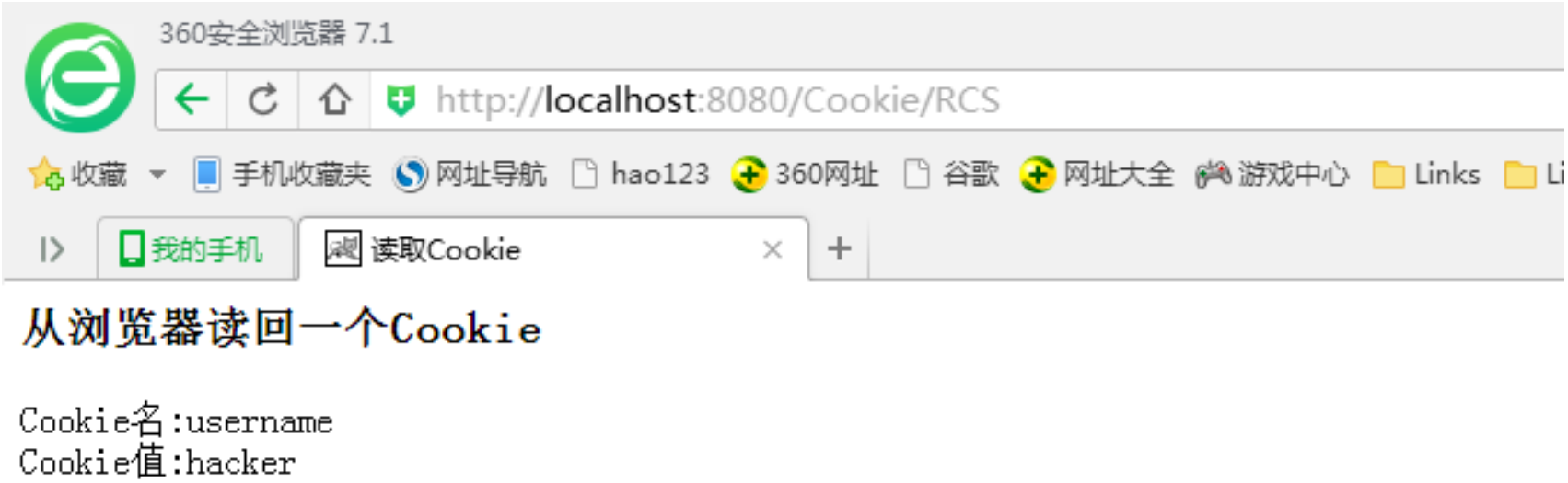
ReadCookieServlet.java

```
package com.demo;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ReadCookieServlet extends HttpServlet{
   public void doGet(HttpServletRequest request,
                              HttpServletResponse response)
        throws IOException,ServletException{
        String cookieName = "username";
        String cookieValue = null;
Cookie[] cookies = request.getCookies();
  if (cookies!=null){
      for(int i = 0;i<cookies.length;i++){
            Cookie cookie = cookies[i];
  if (cookie.getName().equals(cookieName))
            cookieValue = cookie.getValue();
          }
        }
```

ReadCookieServlet.java

```
response.setContentType("text/html;charset=UTF-8");
PrintWriter out=response.getWriter();
out.println("<html><title>读取Cookie</title>");
out.println("<body><h3>从浏览器读回一个Cookie</h3>");
out.println("Cookie名:"+cookieName+"<br>");
out.println("Cookie值:"+cookieValue+"<br>");
out.println("</body></html>");
   }
}
```

运行效果


360安全浏览器 7.1
http://localhost:8080/Cookie/RCS
收藏
手机收藏夹
网址导航
hao123
360网址
谷歌
网址大全
游戏中心
Links
我的手机
读取Cookie
从浏览器读回一个Cookie
Cookie名:username
Cookie值:hacker

3.3 小结

- 在Java Web开发中通过使用HttpSession对象可以跟踪客户与服务器的交互，Web应用程序需要在本来无状态的HTTP协议上实现状态。
- Cookie实际上是服务器发送给客户端的一小段文本信息，客户以后访问同一个Web服务器时浏览器会把它们原样发送给服务器。