第5章 Servlet的会话技术


视频讲解

本章学习目标：

- 能正确描述会话的原理与特点，说明解决 HTTP 缺陷的基本方法。
- 能说明 Cookie 的工作原理，并熟练应用 Cookie 编程。
- 能说明 Session 的工作原理，并熟练应用 Session 编程。
- 熟练应用 URL 重写技术编程。

主要知识点：

- Cookie 的工作原理和基本方法。
- Session 的工作原理和基本方法。
- Cookie 和 Session 的正确应用。

思想引领：

- 介绍 Servlet 会话的原理、特点和安全性。
- 培养学生软件设计的安全意识以及责任担当的胸怀。

在前面章节学习的网络通信通常具有一次性的特点，如第 3 章利用 ServletConfig 与 ServletContext 获取配置文件中的初始参数，在第 4 章利用 HttpServletRequest 请求对象和 HttpServletResponse 响应对象来实现客户端与服务器端的信息传输。但在生活中，为了完成某项任务，通常需要一系列的动作，即反复多次调用请求对象和响应对象的相关方法，有时需要对每个动作的状态进行跟踪，这时必须用到会话技术，本章将重点介绍会话原理及会话技术的应用方法。

5.1 会话技术概述

5.1.1 会话原理与特点

会话(Session)指的是一个客户端(浏览器)与 Web 服务器之间，为了完成某项任务，而连续发生的一系列请求与响应过程。例如，日常生活中的打电话，从拨通电话到挂断电话之间的一连串的你问我答的过程。还有网上购物，从选择商品到结算一系列请求/响应过程。还有 Web 访问，从用户凭密码登录网站，然后在该站点上单击多个超链接访问相关 Web 资源或完成相关活动，直到所有任务完成，关闭浏览器的整个过程，它们都属于一个会话。从以上列举的实例可以看出，会话具有以下两个特点。

(1) 相关性。会话过程产生的多个请求或响应通常是为了完成共同的任务。

(2) 系列性。一个会话过程不可能在一个请求/响应中完成,它通常包含一系列的请求/响应。

5.1.2 HTTP 缺陷的解决方法

刚才说明了会话具有相关性和系列性的特点,所以在会话过程中,同一会话内部的多次请求与响应必然会共享一些数据。但是,HTTP 是无状态的协议,用户一次请求的数据一旦交换完毕,客户端与服务器端的连接就会中断,这些内部共享数据就会丢失,当用户再次请求(如单击其他链接)时,需要建立新的连接,这时服务器无法再次访问丢失的数据,这说明 HTTP 无法从 HTTP 连接上跟踪用户的会话(实现会话技术)。例如,用户甲在某购物网站上选择了一件商品,当去结算商品时,服务器已经忘记了用户甲在前面选择了什么商品。

有些读者可能会说,利用前面章节中介绍的 HttpServletRequest、ServletConfig 和 ServletContext 对象不是可以实现数据的共享与传递吗? 是的,但是它们也无法实现会话技术,原因如下。

(1) HttpServletRequest 对象只能保存本次请求所传递的数据,但会话过程中有一系列不同请求的数据需要保存。

(2) ServletConfig 对象对应的数据只能被一个 Servlet 对象访问,但一个 Servlet 对象无法完成一次会话。

(3) ServletContext 对象对应的共享数据可以被整个 Web 应用的所有会话共享,服务器无法区分相关数据是属于哪个会话的,所以无法保证数据的安全。

为了让服务器能安全地保存每个会话过程中产生的数据,Servlet 提供了 Cookie 和 Session 两个对象,它们弥补了以上缺点,下面详细介绍。

5.2 Cookie 对象的应用

Cookie 是由 W3C 组织提出的一种会话技术,最早由 Netscape 社区发展的一种会话跟踪机制。目前 Cookie 已经成为标准,所有的主流浏览器都支持 Cookie 技术。Cookie 是将会话过程中产生的用户数据保存在用户浏览器中的一种会话技术。它使浏览器记住用户状态,方便与服务器进行数据交互。就像现实生活中商城赠送给顾客的购物卡,卡中记录了用户的姓名、手机号、消费额度和积分数量等信息。当顾客使用该购物卡购物时,商场可以根据购物卡中保存的信息计算会员的优惠额度和累加积分。

5.2.1 Cookie 的工作原理

读者在第 4 章学习了 HTTP 请求头和 HTTP 响应头,Web 就是利用它们来传送 Cookie 信息的,在 HTTP 请求头中定义了 Cookie 头字段,在 HTTP 响应头中定义了 Set-Cookie 头字段,它们分别用于发送和设置 Cookie 信息,其工作原理如图 5-1 所示。

下面来分析 Cookie 的信息传递过程。

(1) 浏览器第一次访问 Server1 时,向 HTTP 服务器发送的 HTTP 请求,其中不包含 Cookie 信息。

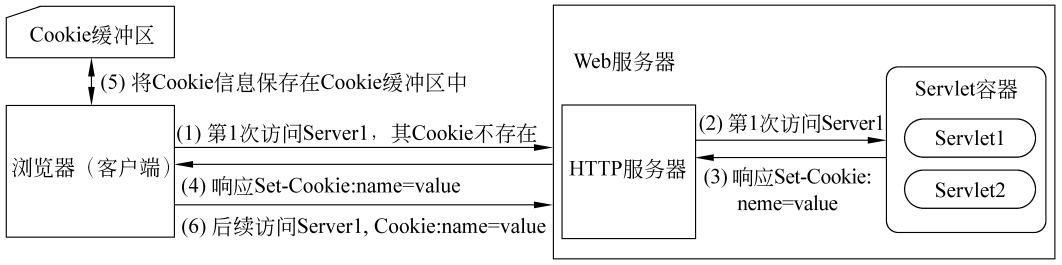


图 5-1　Cookie 的工作原理

（2）HTTP 服务器发现客户的请求是访问动态网页，于是将 Server1 请求转交给 Server 容器。

（3）Server 容器发现浏览器发来的请求是第一次请求，于是按 Set-Cookie：name＝value 格式，将 Cookie 添加到响应报文的首部，然后发给 HTTP 服务器。

（4）HTTP 服务器将 Set-Cookie：name＝value 信息转发给浏览器。

（5）浏览器收到服务器发回的 Set-Cookie 后，会将 Cookie 信息保存在 Cookie 缓冲区中，或者写入 Server1 对应的 Cookie 文件中。

（6）当浏览器再次访问 Server1 时，会从 Cookie 缓存中读取 Server1 的 Cookie 数据，然后添加到 HTTP 请求报文中，再发给服务器。

5.2.2　Cookie 的基本方法

在 Servlet API 提供的 javax.servlet.http 包中包含 Cookie 类，该类包含 Cookie 的构造函数和提取与设置 Cookie 相关属性的方法。

1. Cookie 的构造函数

Cookie 的构造函数用于创建 Cookie 对象，其格式如下。

```
public Cookie(String name,String value)
```

其中，参数 name 指定 Cookie 的名称，参数 value 指定 Cookie 的值。要注意的是，Cookie 一旦被创建，它的名称就不能被修改，但它的值可以被修改。

Cookie 对象创建后，可以利用响应对象 response 的 addCookie()方法将它添加到响应报文中，也可以利用请求对象 request 的 getCookies()方法从请求报文中获取 Cookie 对象，例如：

```
Cookie cookie1 = new Cookie("mybook","JavaWeb 程序设计");
response.addCookie(cookie1);
Cookie[] cookies = request.getCookies();
```

2. Cookie 的常用方法

HTTP 响应报文的 Set-Cookie 头字段中包含 Cookie 的名称与值、Cookie 的有效日期、有效路径、有效域、加密认证协议（即 HTTPS）等信息，其格式如下。

```
Set-Cookie:
    NAME=VALUE;Expires=DATE;Path=PATH;
    Domain=DOMAIN_NAME;SECURE
```

Cookie 对象中包含很多提取和设置这些 Cookie 属性的方法,Cookie 的常用方法如表 5-1 所示。

表 5-1 Cookie 的常用方法

方 法 格 式	功 能 描 述
public String getName()	读取 Cookie 的名字,如: String name = cookie1.getName()
public void setValue(String newValue)	设置 Cookie 的值,如: cookie1.setValue("Python 程序设计")
public String getValue()	读取 Cookie 的值,如: String value =cookie1.getValue()
public void setMaxAge(int expiry)	设置 Cookie 的保存时间,即有效期,以 s 为单位。如果设置为正数,则 Cookie 保存在本地硬盘中规定的时间;如果设置为 0,则立即删除 Cookie;如果设置为负数,则 Cookie 保存在缓存中,关闭浏览器时 Cookie 被删除,默认值是 -1。例如: cookie1.setMaxAge(60 * 60 * 24)
public int getMaxAge()	读取 Cookie 的保存时间,如: int remain= cookie1.getMaxAge()
public void setPath(String url)	设置 Cookie 的有效路径,默认值是当前路径,即产生 Cookie 的 Servlet 对应的 url。如: cookie1. setPath("/")是整个网站的目录
public String getPath()	获取 Cookie 的有效路径。如: String myPath =cookie1.getPath()
public void setDomain(String pattern)	设置 Cookie 的有效域,域名 pattern 必须以点(".")开始,默认值为当前主机名。如: cookie1. setDomain(".www.sgu.com.cn")
public String getDomain()	获取 Cookie 的有效域,domain 属性的值是不区分大小写的。如: String myDomain =cookie1.getDomain()
public void setSecure(boolean flag)	设置 Cookie 是否使用加密认证协议
public boolean getSecure()	返回 Cookie 是否使用加密认证协议

注意: Cookie 可以在同一浏览器的不同窗口间共享数据,但不能在不同浏览器间共享数据。另外,Cookie 是不可跨域访问的,也就是说,在 Server1 服务器上发布的 Cookie 不会被提交到 Server2 服务器上,这是由 Cookie 的隐私安全机制决定的。接下来设计一个在浏览器端保存用户名的 Cookie 应用实例,如例 5-1 所示。

【例 5-1】 在浏览器端保存用户名的 Cookie 应用实例,设计过程如下。

第 1 步,在 MyEclipse 平台新建 Web5CookieSession 项目,在项目的 WebRoot 目录下创建表单输入页 n501Login.html,代码如下。

```
<!DOCTYPE html>
<html>
<head>
  <title>n501Login.html</title>
  <meta name="content-type" content="text/html; charset=gb2312">
</head>
<body>
  <form name="reg" action="servlet/N501LoginCookie" method="post">
      <h3>用户登录窗口</h3>
```

```
    用户名: <input name="myName" type="text" /><br/>
    <input type="submit" value="提交" />
    <input type="reset"  value="重置">
  </form>
</body>
</html>
```

第 2 步,在该项目下的 src 目录中创建 ch5 包,在 ch5 包中创建以下 Servlet 代码。

```
package ch5;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class N501LoginCookie extends HttpServlet {
        public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out = response.getWriter();
        request.setCharacterEncoding("gb2312");
        //获取本次登录的用户名
        String username = request.getParameter("myName");
        String lastname = null;                          //用于保存上次登录的用户名
        Cookie[] cookies = request.getCookies();
        //遍历 cookies 数组,查找名称为 userInfo 的 Cookie 的值
        for (int i = 0; cookies != null && i < cookies.length; i++)
        {
            if ("userInfo".equals(cookies[i].getName())) {
                lastname = cookies[i].getValue();
                break;
            }
        }
        //如果找到了,则输出两次登录的用户名
        if (lastname != null) {
            out.print("<br>您上次登录的用户名是:"+lastname);
            out.print("<br>您本次登录的用户名是:"+username);
        } else {
            out.print("<br>您是首次访问本站!!!");
            out.print("<br>您本次登录的用户名是:"+username);
        }
        //创建名称为 userInfo 的 cookie,保存当前用户信息
        Cookie cookie = new Cookie("userInfo",username);
        //cookie.setMaxAge(60*60);                       //设置 cookie 最大存在时间
        //发送 cookie
        response.addCookie(cookie);
        out.flush();
```

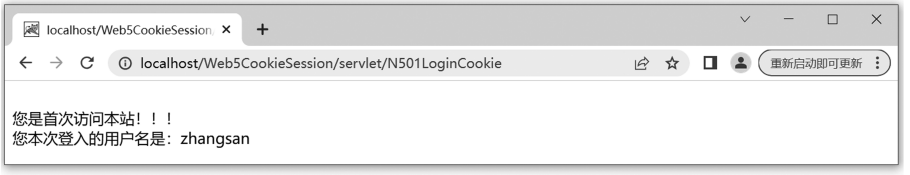
```
            out.close();
        }
        public void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
            this.doGet(request, response);
        }
}
```

第 3 步,测试以上代码。在浏览器中输入网址:

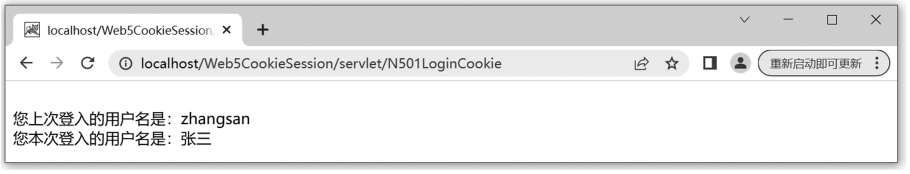
```
http://localhost/Web5CookieSession/n501Login.html
```

网页运行结果如图 5-2 所示。

(a) 输入表单显示的结果



(b) 第1次输入用户名“提交”后的结果



(c) 第2次输入用户名“提交”后的结果

图 5-2　网页运行结果

5.3　Session 对象的应用

前面介绍的 Cookie 虽然可以实现会话技术,但 Cookie 是将用户的信息保存在各自的浏览器中的,这些信息需要传送到服务器端处理,当传递的信息比较多时,会增加网络带宽的负担和服务器程序的处理难度。为了解决以上问题,提出了一种 Session 技术,它将会话过程中产生的用户数据保存在服务器端,客户端只需保存 Session 的标识(即 Session ID)。就像大学内保存的学生档案、医院内保存的病人病历、公安局户政科保存的公民户口等数

据，其信息量都比较大，不方便随身携带。如果将它们保存在各个单位的数据库中，那么每个学生或病人或公民只需携带学生证（上面有学生证号）或就诊卡（有卡号）或身份证（有身份证号），服务器通过相关编号就可以查到数据库中保存的详细信息。

5.3.1 Session 的工作原理

与 Cookie 一样，Session 也是用来解决 HTTP 无状态的问题。不同的是，它采用服务器端的工作机制，用类似于散列表的数据结构将会话过程中产生的数据保存在服务器端。Session 的工作原理如图 5-3 所示。

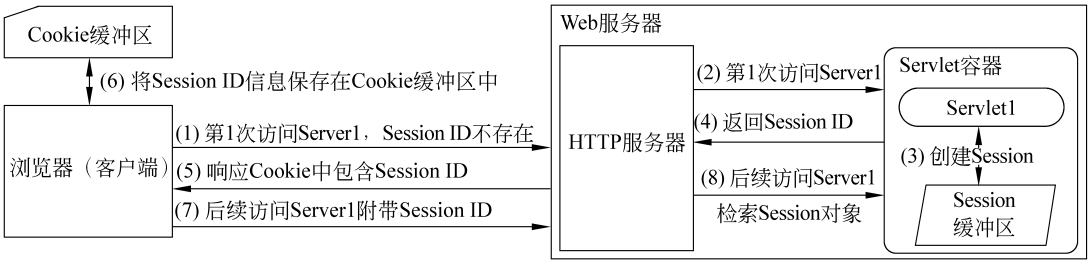


图 5-3 Session 的工作原理

下面分析 Session 的信息传递过程。

（1）浏览器第一次访问 Server1 时，向 HTTP 服务器发送 HTTP 请求，其中不包含 Session ID 信息。

（2）HTTP 服务器发现客户的请求是访问动态网页，于是将 Server1 请求转交给 Server 容器。

（3）Server 容器发现浏览器发来的请求是第一次请求，于是为客户创建一个 Session 对象放在缓冲区中。

（4）Server 容器将 Session ID 信息返回给 HTTP 服务器。

（5）HTTP 服务器按 Set-Cookie:name=value 格式，将包含 Session ID 的 Cookie 添加到响应报文的首部，然后发给浏览器。

（6）浏览器收到 HTPP 服务器发回的 Session ID 后，将 Session ID 信息保存在 Cookie 缓冲区中，或者写入 Server1 对应的 Cookie 文件中。

（7）当浏览器再次访问 Server1 时，会从 Cookie 缓存中读取 Server1 的 Session ID 数据，然后添加到 HTTP 请求报文中，一起发给服务器。

（8）服务器就按照收到的 Session ID 把 Session 对象从 Cookie 缓存中检索出来使用。当然，如果没有检索到客户端提交的 Session ID，说明该 Session 已经被注销或超时，服务器会新建一个 Session 对象。

5.3.2 Session 的常用方法

Servlet API 提供的 javax.servlet.http 包中包含 Session 类，该类用于会话管理，可以通过前面介绍的 HttpServletRequest 接口中的以下两个方法获取 Session 对象。

（1）public HttpSession getSession(boolean create)：返回与此次请求相关联的 HttpSession 对象。如果不存在，则根据参数 create 的值是 true 或 false 来决定是创建一个

新的 HttpSession 返回,还是返回 null 值。

(2) public HttpSession getSession(): 其功能等同于 getSession(true),即返回与此次请求相关联的 HttpSession 对象。如果不存在,则创建一个新的 HttpSession 返回。

HttpSession 对象的常用方法如表 5-2 所示。

表 5-2 HttpSession 对象的常用方法

方法格式	功能描述
public String getId()	返回 Session 对象的唯一标识符
public Enumeration getAttributeNames()	返回 Session 对象中的所有属性名的枚举集合
public Object getAttribute(String name)	返回 Session 对象中属性名为 name 的属性值
public void setAttribute(String name, Object value)	设置 Session 对象中的属性名对应的属性值
public void removeAttribute(String name)	删除 Session 对象中名称为 name 的属性
public long getCreationTime()	返回以 ms 为单位的 Session 对象的创建时间
public long getLastAccessedTime()	返回以 ms 为单位的 Session 对象的最后一次被访问的时间
public int getMaxInactiveInterval()	返回以 ms 为单位的 Session 对象的有效期,超过这个有效期,Servlet 容器将使 Session 失效
public void setMaxInactiveInterval(int interval)	以 ms 为单位设置在 Session 对象的有效期。如果参数为负数,则该 Session 对象永不失效
public ServletContext getServletContext()	返回 Session 对象所属的 ServletContext 对象
public void invalidate()	立刻摧毁 Session 对象
public boolean isNew()	判断当前 Session 对象是否是新建的

另外,Session 对象的有效期也可以在 web.xml 文件中设置,该 web.xml 配置文件对站点内的所有 Web 应用程序都起作用,不过该设置的时间单位是分钟,不是 ms。在 Tomcat 的安装目录的 conf\web.xml 文件中,可以找到以下配置信息。

```
<session-config>
    <session-timeout>30 </session-timeout>
</session-config>
```

以上代码设置了 Session 对象的有效期为 30min。如果将<session-timeout>元素中的时间值设置成 0 或负数,则表示 Session 永不超时。另外,如果想单独设置某个 Web 应用程序的 Session 有效期,则需要在自己 Web 应用的 web.xml 文件中进行设置。接下来设计一个 Session 在网站购物车中的应用实例,如例 5-2 所示。

【例 5-2】 Session 在网站购物车中的应用实例,设计过程如下。

第 1 步,在 ch5.n502cartJava 子包中创建图书信息类 Book,其 Java 代码如下。

```
package ch5.n502cartJava;
import java.io.Serializable;
public class Book implements Serializable {
    private int id;                    //编号
```

```
    private String name;                //书名
    private double price;               //售价
    private String author;              //作者
    public int getId() {  return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public double getPrice() { return price; }
    public void setPrice(double price) { this.price = price; }
    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }
}
```

第 2 步，在 ch5.n502cartJava 子包中创建 BookDB 类模拟数据库，其 Java 代码如下。

```
package ch5.n502cartJava;
import java.util.ArrayList;
public class BookDB {
    public static ArrayList<Book> books = new ArrayList<Book>();
    static{
        Book b1 = new Book();
        b1.setId(0);
        b1.setName("唐诗三百首");
        b1.setAuthor("(清)蘅塘居士");
        b1.setPrice(24);
        Book b2 = new Book();
        b2.setId(1);
        b2.setName("宋词三百首");
        b2.setAuthor("(清)朱孝臧");
        b2.setPrice(29.8);
        Book b3 = new Book();
        b3.setId(2);
        b3.setName("笠翁对韵");
        b3.setAuthor("(明末清初)李渔");
        b3.setPrice(32);
        books.add(b1);
        books.add(b2);
        books.add(b3);
    }
    public static ArrayList<Book> getAllBooks(){
        return books;
    }
    public static Book getBookById(int id){
        return books.get(id);
    }
}
```

第 3 步,在 ch5.n502cartJava 子包中创建图书显示类 BooksList,其代码如下。

```
package ch5.n502cartJava;
import java.io.*;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class BooksList extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        ArrayList<Book> books = BookDB.getAllBooks();
        out.print("本站提供的图书有:<br>");
        for (Book book : books) {
            //定义教材的选购链接
            String url = "BooksBuy?id=" + book.getId();
            out.print("<br>书名:"+book.getName()+",售价:"
            +book.getPrice() + ",作者:"+book.getAuthor());
            out.print("<a href='" + url+ "'>\t 放入购物车</a>
            <br>");
        }
    }
    public void doPost(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        this.doGet(request, response);
    }
}
```

第 4 步,在 ch5.n502cartJava 子包中创建图书购买类 BooksBuy,其代码如下。

```
package ch5.n502cartJava;
import java.io.*;
import java.util.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class BooksBuy extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        //获得用户购买的商品
        String id = request.getParameter("id");
```