# 第5章

1. 填空
2. Set-Cookie
3. 服务器
4. 服务器
5. Cookie
6. web.xml
7. 判断
8. 对
9. 对
10. 错
11. 错
12. 错
13. 选择
14. C
15. B
16. D
17. C
18. B
19. 简答

会话过程类似于生活中的打电话过程，它指的是一个客户端（浏览器）与Web服务器之间连续发生的一系列请求和响应过程。在Servlet技术中，提供了两个用于保存会话数据的对象，分别是Cookie和Session。

1.    Cookie和HttpSession是保存会话相关数据的技术，其中Cookie将信息存储在浏览器端是客户端技术，Session将数据保存在服务器端是服务器端技术。

2.    Cookie是基于HTTP协议中的Set-Cookie响应头和Cookie请求头进行工作的。

3.    默认情况下HttpSession是基于一个名称为JSESSIONID 的特殊Cookie工作的。

4.    浏览器对Cookie具有严格的限制，一个网站能在浏览器中保存多少Cookie是有限制的。

5.    HttpSession默认是基于Cookie运作的。

1. 编程

1.

public class PurchaseServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

// 获得用户购买的商品

String id = req.getParameter("id");

if (id == null) {

// 如果id为null，重定向到ListBookServlet页面

String url = "/ListBookServlet";

resp.sendRedirect(url);

return;

}

Book book = BookDB.getBook(id);

// 创建或者获得用户的Session对象

HttpSession session = req.getSession();

// 从Session对象中获得用户的购物车

List<Book> cart = (List)session.getAttribute("cart");

if (cart == null) {

// 首次购买，为用户创建一个购物车(List集合模拟购物车)

cart = new ArrayList<Book>();

// 将购物城存入Session对象

session.setAttribute("cart", cart);

}

// 将商品放入购物车

cart.add(book);

// 创建Cookie存放Session的标识号

Cookie cookie = new Cookie("JSESSIONID", session.getId());

cookie.setMaxAge(60 \* 30);

cookie.setPath("/chapter06");

resp.addCookie(cookie);

// 重定向到购物车页面

String url = "BookServlet";

resp.sendRedirect(url);

}

}

2.

LastAccessServlet类的实现代码如下：

public class LastAccessServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

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

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

/\*

\* 设定一个 cookie 的name : lastAccessTime 读取客户端发送cookie

\* 获得用户上次的访问时间显示

\*/

String lastAccessTime = null;

// 获取所有的cookie，并将这些cookie存放在数组中

Cookie[] cookies = request.getCookies();

for (int i = 0; cookies != null && i < cookies.length; i++) {

if ("lastAccess".equals(cookies[i].getName())) {

// 如果cookie的名称为lastAccess,则获取该cookie的值

lastAccessTime = cookies[i].getValue();

break;

}

}

// 判断是否存在名称为lastAccess的cookie

if (lastAccessTime == null) {

response.getWriter().print("您是首次访问本站！！！");

} else {

response.getWriter().print("您上次的访问时间" + lastAccessTime);

}

// 创建cookie,将当前时间作为cookie的值发送给客户端

Cookie cookie = new Cookie("lastAccess", new Date().toLocaleString());

cookie.setMaxAge(60 \* 60);// 保存1小时

// 访问chapter05下资源时回送cookie

cookie.setPath("/chapter05");

// 发送 cookie

response.addCookie(cookie);

}

}