# 第5章

1. 填空
2. 服务器向客户端发送Cookie时，会在HTTP响应头字段中增加\_\_\_\_响应头字段。
3. Web应用中的会话指的是一个客户端（浏览器）与\_\_\_\_之间连续发生的一系列请求和响应过程。
4. Session相对于Cookie具有更高的安全性，它将关键数据保存在\_\_\_\_端。
5. 在Servlet技术中，提供了两个用于保存会话数据的对象，分别是\_\_\_\_和Session。
6. Tomcat容器中会话的有效时间可以在\_\_\_\_文件中设置，默认会话过期时间为30分钟。
7. 判断
8. Tomcat容器中如果将元素中的时间值设置成0或一个负数，则表示会话永不超时。 T
9. Session是一种将会话数据保存到服务器端的技术，需要借助Cookie技术来实现。T
10. Cookie的domain属性是用来指定浏览器访问的域，设置domain属性时严格区分大小写。F(?)
11. Cookie的path属性设置后，只对当前访问路径所属的目录有效。F
12. 一次会话过程中，只允许客户端与服务器进行一次请求与响应过程。F
13. 选择
14. 下列对于setMaxAge(-1)方法的描述中，正确的是（）C

A: 表示通知浏览器保存这个Cookie信息

B: 表示通知浏览器立即删除这个Cookie信息

C: 表示当浏览器关闭时，Cookie信息会被删除

D: 以上说法都是错误的

1. Tomcat服务器的默认会话超时时间是（）B

A: 30秒

B: 30分钟

C: 30毫秒

D: 30小时

1. 下面选项中，用于强制使Session对象无效的方法是（）?--D

A: request. invalidate ();

B: session. validate ();

C: response. invalidate ();

D: session. invalidate ();

1. 下面选项中，当存在Session对象直接返回，否则返回null的方法是（）?--C

A: request.getSession();

B: request.getSession(true);

C: request.getSession(false);

D: response.getSession();

1. 下列选项中，能够用于获取客户端所有cookie对象的方法是（）B

A: List cookies = request.getCookies();

B: Cookie[] cookies = request.getCookies();

C: List cookies = response.getCookies();

D: Cookie[] cookies = response.getCookies();

1. 简答
2. 简述什么是会话技术？

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. 简述Cookie与Session的区别。（至少写出3点）

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
2. 请按照以下要求设计一个购物车的类PurchaseServlet。

书籍数据如下：

public class Book {

private static final long serialVersionUID = 1L;

private String id;

private String name;

// getter/setter方法省略

public Book(String id, String name) {

this.id = id;

this.name = name;

}

}

public class BookDB {

private static Map<String, Book> books = new LinkedHashMap<String, Book>();

static {

books.put("1", new Book("1", "javaweb开发"));

books.put("2", new Book("2", "jdbc开发"));

books.put("3", new Book("3", "java基础"));

books.put("4", new Book("4", "struts开发"));

books.put("5", new Book("5", "spring开发"));

}

// 获得所有的图书

public static Collection<Book> getAll() {

return books.values();

}

// 根据指定的id获得图书

public static Book getBook(String id) {

return books.get(id);

}

}

ListBookServlet.java代码：

@WebServlet(name = "ListBookServlet", urlPatterns = "/ListBookServlet")

public class ListBookServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

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

PrintWriter out = resp.getWriter();

Collection<Book> books = BookDB.getAll();

out.write("本站提供的书籍有：<br>");

for (Book book : books) {

String url = "PurchaseServlet?id=" + book.getId();

out.write(book.getName() + "<a href='" + url

+ "'>点击购买</a><br>");

}

}

}

BookServlet.java代码：

@WebServlet(name = "BookServlet", urlPatterns = "/BookServlet")

public class BookServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException {

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

PrintWriter out = resp.getWriter();

// 变量cart引用用户的购物车

List<Book> cart = null;

// 变量purFlag标记用户是否买过商品

boolean purFlag = true;

// 获得用户的session

HttpSession session = req.getSession(false);

// 如果session为null，purFlag置为false

if (session == null) {

purFlag = false;

} else {

// 获得用户购物车

cart = (List) session.getAttribute("cart");

// 如果用的购物车为null，purFlag置为false

if (cart == null) {

purFlag = false;

}

}

/\*

\* 如果purFlag为false，表明用户没有购买蛋糕 重定向到ListServlet页面

\*/

if (!purFlag) {

out.write("对不起！您还没有购买任何商品！<br>");

} else {

// 否则显示用户购买蛋糕的信息

out.write("您购买的书籍有：<br>");

double price = 0;

for (Book book : cart) {

out.write(book.getName() + "<br>");

}

}

}

}

要求如下：

1)    创建一个PurchaseServlet类，使其继承HttpServlet类并重写该类的doGet()方法；

2)    在doGet()方法中，使用request.getSession()实现购物车；

3)    将session的id保存到cookie中；

4)    重定向到BookServlet。

2、设计一个类，使用Cookie技术实现显示用户上次访问时间的功能。

（1）创建一个LastAccessServlet类，使其继承HttpServlet类并重写该类的doGet()方法。

（2）在doGet()方法中，使用request.getCookies()得到所有cookie形成的cookie数组，并进行遍历。

（3）如果遍历过程中找到cookie信息中存在lastAccess属性则输出，否则创建cookie对象设置值为当前时间并发送给客户端。

（4）cookie的存活时间为1小时，访问当前应用中的所有资源客户端都回送cookie信息。