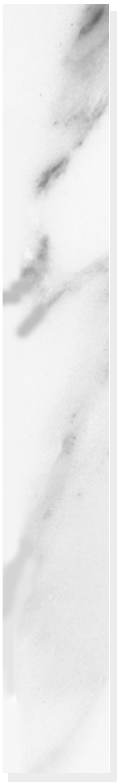
© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

HANDLING COOKIES



Topics in This Chapter

- Purposes for cookies
- Problems with cookies
- The `Cookie` API
- A simple servlet that sets cookies
- A cookie-reporting servlet
- Some utilities that simplify cookie handling
- A customized search engine front end based upon cookies

Online version of this first edition of ***Core Servlets and JavaServer Pages*** is free for personal use. For more information, please see:

- **Second edition of the book**: http://www.coreservlets.com.
- **Sequel**: http://www.moreservlets.com.
- **Servlet and JSP training courses from the author**: http://courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Chapter 8

Cookies are small bits of textual information that a Web server sends to a browser and that the browser returns unchanged when later visiting the same Web site or domain. By letting the server read information it sent the client previously, the site can provide visitors with a number of conveniences such as presenting the site the way the visitor previously customized it or letting identifiable visitors in without their having to enter a password. Most browsers avoid caching documents associated with cookies, so the site can return different content each time.

This chapter discusses how to explicitly set and read cookies from within servlets, and the next chapter shows you how to use the servlet session tracking API (which can use cookies behind the scenes) to keep track of users as they move around to different pages within your site.

8.1 Benefits of Cookies

This section summarizes four typical ways in which cookies can add value to your site.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Identifying a User During an E-commerce Session

Many on-line stores use a "shopping cart" metaphor in which the user selects an item, adds it to his shopping cart, then continues shopping. Since the HTTP connection is usually closed after each page is sent, when the user selects a new item to add to the cart, how does the store know that it is the same user that put the previous item in the cart? Persistent (keep-alive) HTTP connections (see Section 7.4) do not solve this problem, since persistent connections generally apply only to requests made very close together in time, as when a browser asks for the images associated with a Web page. Besides, many servers and browsers lack support for persistent connections. Cookies, however, *can* solve this problem. In fact, this capability is so useful that servlets have an API specifically for session tracking, and servlet authors don't need to manipulate cookies directly to take advantage of it. Session tracking is discussed in Chapter 9.

Avoiding Username and Password

Many large sites require you to register in order to use their services, but it is inconvenient to remember and enter the username and password each time you visit. Cookies are a good alternative for low-security sites. When a user registers, a cookie containing a unique user ID is sent to him. When the client reconnects at a later date, the user ID is returned, the server looks it up, determines it belongs to a registered user, and permits access without an explicit username and password. The site may also remember the user's address, credit card number, and so forth, thus simplifying later transactions.

Customizing a Site

Many "portal" sites let you customize the look of the main page. They might let you pick which weather report you want to see, what stock and sports results you care about, how search results should be displayed, and so forth. Since it would be inconvenient for you to have to set up your page each time you visit their site, they use cookies to remember what you wanted. For simple settings, this customization could be accomplished by storing the page settings directly in the cookies. Section 8.6 gives an example of this. For more complex customization, however, the site just sends the client a unique identifier and keeps a server-side database that associates identifiers with page settings.

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Focusing Advertising

Most advertiser-funded Web sites charge their advertisers much more for displaying "directed" ads than "random" ads. Advertisers are generally willing to pay much more to have their ads shown to people that are known to have some interest in the general product category. For example, if you go to a search engine and do a search on "Java Servlets," the search site can charge an advertiser much more for showing you an ad for a servlet development environment than for an ad for an on-line travel agent specializing in Indonesia. On the other hand, if the search had been for "Java Hotels," the situation would be reversed. Without cookies, the sites have to show a random ad when you first arrive and haven't yet performed a search, as well as when you search on something that doesn't match any ad categories. Cookies let them remember "Oh, that's the person who was searching for such and such previously" and display an appropriate (read "high priced") ad instead of a random (read "cheap") one.

8.2 Some Problems with Cookies

Providing convenience to the user and added value to the site owner is the purpose behind cookies. And despite much misinformation, cookies are not a serious security threat. Cookies are never interpreted or executed in any way and thus cannot be used to insert viruses or attack your system. Furthermore, since browsers generally only accept 20 cookies per site and 300 cookies total and since each cookie can be limited to 4 kilobytes, cookies cannot be used to fill up someone's disk or launch other denial of service attacks.

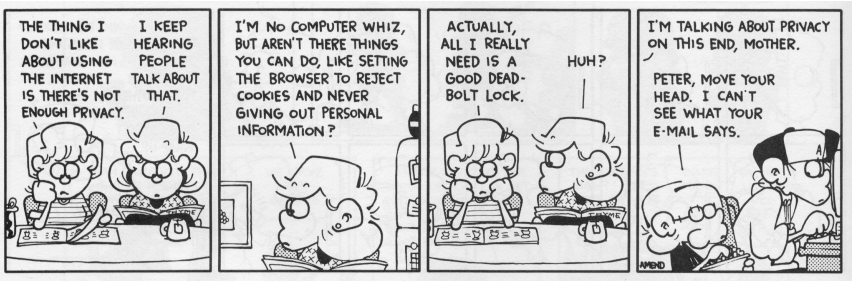
However, even though cookies don't present a serious *security* threat, they can present a significant threat to *privacy*. First, some people don't like the fact that search engines can remember that they're the user who usually does searches on certain topics. For example, they might search for job openings or sensitive health data and don't want some banner ad tipping off their coworkers next time they do a search. Even worse, two sites can share data on a user by each loading small images off the same third-party site, where that third party uses cookies and shares the data with both original sites. (Netscape, however, provides a nice feature that lets you refuse cookies from sites other than that to which you connected, but without disabling cookies altogether.) This trick of associating cookies with images can even be exploited via e-mail if you use an HTML-enabled e-mail reader that "sup-

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

ports" cookies and is associated with a browser. Thus, people could send you e-mail that loads images, attach cookies to those images, then identify you (e-mail address and all) if you subsequently visit their Web site. Boo.

A second privacy problem occurs when sites rely on cookies for overly sensitive data. For example, some of the big on-line bookstores use cookies to remember users and let you order without reentering much of your personal information. This is not a particular problem since they don't actually display the full credit card number and only let you send books to an address that was specified when you *did* enter the credit card in full or use the username and password. As a result, someone using your computer (or stealing your cookie file) could do no more harm than sending a big book order to your address, where the order could be refused. However, other companies might not be so careful, and an attacker who got access to someone's computer or cookie file could get on-line access to valuable personal information. Even worse, incompetent sites might embed credit card or other sensitive information directly in the cookies themselves, rather than using innocuous identifiers that are only linked to real users on the server. This is dangerous, since most users don't view leaving their computer unattended in their office as being tantamount to leaving their credit card sitting on their desk.



FOXTROT © 1998 Bill Amend. Reprinted with permission of UNIVERSAL PRESS SYNDICATE. All rights reserved

The point of all this is twofold. First, due to real and perceived privacy problems, some users turn off cookies. So, even when you use cookies to give added value to a site, your site shouldn't *depend* on them. Second, as the author of servlets that use cookies, you should be careful not to use cookies for particularly sensitive information, since this would open users up to risks if somebody accessed their computer or cookie files.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

8.3 The Servlet Cookie API

To send cookies to the client, a servlet should create one or more cookies with designated names and values with `new Cookie(name, value)`, set any optional attributes with `cookie.setXxx` (readable later by `cookie.getXxx`), and insert the cookies into the response headers with `response.addCookie(cookie)`. To read incoming cookies, a servlet should call `request.getCookies`, which returns an array of `Cookie` objects corresponding to the cookies the browser has associated with your site (this is `null` if there are no cookies in the request). In most cases, the servlet loops down this array until it finds the one whose name (`getName`) matches the name it had in mind, then calls `getValue` on that `Cookie` to see the value associated with that name. Each of these topics is discussed in more detail in the following sections.

Creating Cookies

You create a cookie by calling the `Cookie` constructor, which takes two strings: the cookie name and the cookie value. Neither the name nor the value should contain white space or any of the following characters:

```
[ ] ( ) = , " / ? @ : ;
```

Cookie Attributes

Before adding the cookie to the outgoing headers, you can set various characteristics of the cookie by using one of the following `setXxx` methods, where `Xxx` is the name of the attribute you want to specify. Each `setXxx` method has a corresponding `getXxx` method to retrieve the attribute value. Except for name and value, the cookie attributes apply only to *outgoing* cookies from the server to the client; they aren't set on cookies that come *from* the browser to the server. See Appendix A (Servlet and JSP Quick Reference) for a summarized version of this information.

public String getComment()
public void setComment(String comment)

These methods look up or specify a comment associated with the cookie. With version 0 cookies (see the upcoming subsection on

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

getVersion and setVersion), the comment is used purely for informational purposes on the server; it is not sent to the client.

public String getDomain()
public void setDomain(String domainPattern)

These methods get or set the domain to which the cookie applies. Normally, the browser only returns cookies to the exact same hostname that sent them. You can use setDomain method to instruct the browser to return them to other hosts within the same domain. To prevent servers setting cookies that apply to hosts outside their domain, the domain specified is required to start with a dot (e.g., .prenhall.com), and must contain two dots for noncountry domains like .com, .edu and .gov; and three dots for country domains like .co.uk and .edu.es. For instance, cookies sent from a servlet at bali.vacations.com would not normally get sent by the browser to pages at mexico.vacations.com. If the site wanted this to happen, the servlets could specify cookie.setDomain(".vacations.com").

public int getMaxAge()
public void setMaxAge(int lifetime)

These methods tell how much time (in seconds) should elapse before the cookie expires. A negative value, which is the default, indicates that the cookie will last only for the current session (i.e., until the user quits the browser) and will not be stored on disk. See the LongLivedCookie class (Listing 8.4), which defines a subclass of Cookie with a maximum age automatically set one year in the future. Specifying a value of 0 instructs the browser to delete the cookie.

public String getName()
public void setName(String cookieName)

This pair of methods gets or sets the name of the cookie. The name and the value are the two pieces you virtually *always* care about. However, since the name is supplied to the Cookie constructor, you rarely need to call setName. On the other hand, getName is used on almost every cookie received on the server. Since the getCookies method of HttpServletRequest returns an array of Cookie objects, it is common to loop down this array, calling getName until you have a particular name, then check the value with getValue. For an encapsulation of this process, see the getCookieValue method shown in Listing 8.3.

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

public String getPath()
public void setPath(String path)

These methods get or set the path to which the cookie applies. If you don't specify a path, the browser returns the cookie only to URLs in or below the directory containing the page that sent the cookie. For example, if the server sent the cookie from `http://ecommerce.site.com/toys/specials.html`, the browser would send the cookie back when connecting to `http://ecommerce.site.com/toys/bikes/beginners.html`, but not to `http://ecommerce.site.com/cds/classical.html`. The `setPath` method can be used to specify something more general. For example, `someCookie.setPath("/")` specifies that *all* pages on the server should receive the cookie. The path specified must include the current page; that is, you may specify a more general path than the default, but not a more specific one. So, for example, a servlet at `http://host/store/cust-service/request` could specify a path of `/store/` (since `/store/` includes `/store/cust-service/`) but not a path of `/store/cust-service/returns/` (since this directory does not include `/store/cust-service/`).

public boolean getSecure()
public void setSecure(boolean secureFlag)

This pair of methods gets or sets the boolean value indicating whether the cookie should only be sent over encrypted (i.e., SSL) connections. The default is `false`; the cookie should apply to all connections.

public String getValue()
public void setValue(String cookieValue)

The `getValue` method looks up the value associated with the cookie; the `setValue` method specifies it. Again, the name and the value are the two parts of a cookie that you almost *always* care about, although in a few cases, a name is used as a boolean flag and its value is ignored (i.e., the existence of a cookie with the designated name is all that matters).

public int getVersion()
public void setVersion(int version)

These methods get/set the cookie protocol version the cookie complies with. Version 0, the default, follows the original Netscape specification (`http://www.netscape.com/newsref/std/cookie_spec.html`). Version 1, not yet widely supported, adheres to RFC 2109 (retrieve RFCs from the archive sites listed at `http://www.rfc-editor.org/`).

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Placing Cookies in the Response Headers

The cookie is inserted into a `Set-Cookie` HTTP response header by means of the `addCookie` method of `HttpServletResponse`. The method is called `addCookie`, not `setCookie`, because any previously specified `Set-Cookie` headers are left alone and a new header is set. Here's an example:

```
Cookie userCookie = new Cookie("user", "uid1234");
userCookie.setMaxAge(60*60*24*365); // 1 year
response.addCookie(userCookie);
```

Reading Cookies from the Client

To send cookies *to* the client, you create a `Cookie`, then use `addCookie` to send a `Set-Cookie` HTTP response header. To read the cookies that come back *from* the client, you call `getCookies` on the `HttpServletRequest`. This call returns an array of `Cookie` objects corresponding to the values that came in on the `Cookie` HTTP request header. If there are no cookies in the request, `getCookies` returns `null`. Once you have this array, you typically loop down it, calling `getName` on each `Cookie` until you find one matching the name you have in mind. You then call `getValue` on the matching `Cookie` and finish with some processing specific to the resultant value. This is such a common process that Section 8.5 presents two utilities that simplify retrieving a cookie or cookie value that matches a designated cookie name.

8.4 Examples of Setting and Reading Cookies

Listing 8.1 and Figure 8–1 show the `SetCookies` servlet, a servlet that sets six cookies. Three have the default expiration date, meaning that they should apply only until the user next restarts the browser. The other three use `setMaxAge` to stipulate that they should apply for the next hour, regardless of whether the user restarts the browser or reboots the computer to initiate a new browsing session.

Listing 8.2 shows a servlet that creates a table of all the cookies sent to it in the request. Figure 8–2 shows this servlet immediately after the `SetCookies` servlet is visited. Figure 8–3 shows it after `SetCookies` is visited then the browser is closed and restarted.

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.1 SetCookies.java

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Sets six cookies: three that apply only to the current
 *  session (regardless of how long that session lasts)
 *  and three that persist for an hour (regardless of
 *  whether the browser is restarted).
 */

public class SetCookies extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    for(int i=0; i<3; i++) {
      // Default maxAge is -1, indicating cookie
      // applies only to current browsing session.
      Cookie cookie = new Cookie("Session-Cookie " + i,
                                 "Cookie-Value-S" + i);
      response.addCookie(cookie);
      cookie = new Cookie("Persistent-Cookie " + i,
                          "Cookie-Value-P" + i);
      // Cookie is valid for an hour, regardless of whether
      // user quits browser, reboots computer, or whatever.
      cookie.setMaxAge(3600);
      response.addCookie(cookie);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Setting Cookies";
    out.println
      (ServletUtilities.headWithTitle(title) +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +
       "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
       "There are six cookies associated with this page.\n" +
       "To see them, visit the\n" +
       "<A HREF=\"/servlet/coreservlets.ShowCookies\">\n" +
       "<CODE>ShowCookies</CODE> servlet</A>.\n" +
       "<P>\n" +
       "Three of the cookies are associated only with the\n" +
       "current session, while three are persistent.\n" +
       "Quit the browser, restart, and return to the\n" +
       "<CODE>ShowCookies</CODE> servlet to verify that\n" +
       "the three long-lived ones persist across sessions.\n" +
       "</BODY></HTML>");
  }
}
```

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

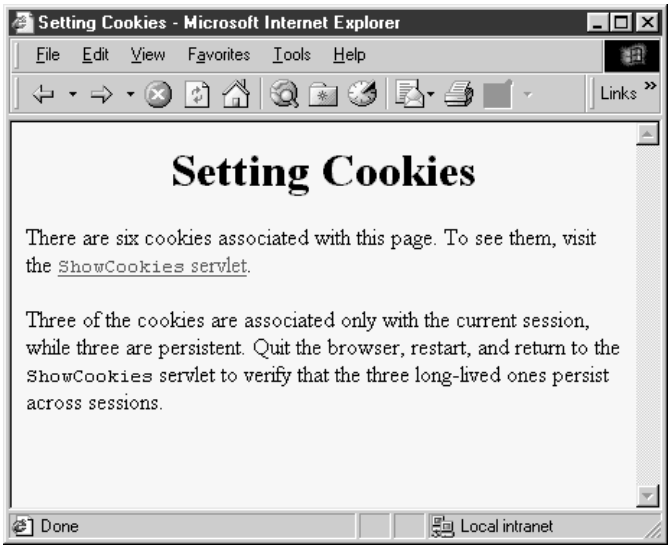


Figure 8–1 Result of `SetCookies` servlet.

Listing 8.2 `ShowCookies.java`

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Creates a table of the cookies associated with
 *  the current page.
 */

public class ShowCookies extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Active Cookies";
    out.println(ServletUtilities.headWithTitle(title) +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
```

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.2 `ShowCookies.java` (continued)

```
                "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
                "<TR BGCOLOR=\"#FFAD00\">\n" +
                "  <TH>Cookie Name\n" +
                "  <TH>Cookie Value");
    Cookie[] cookies = request.getCookies();
    Cookie cookie;
    for(int i=0; i<cookies.length; i++) {
      cookie = cookies[i];
      out.println("<TR>\n" +
                  "  <TD>" + cookie.getName() + "\n" +
                  "  <TD>" + cookie.getValue());
    }
    out.println("</TABLE></BODY></HTML>");
  }
}
```

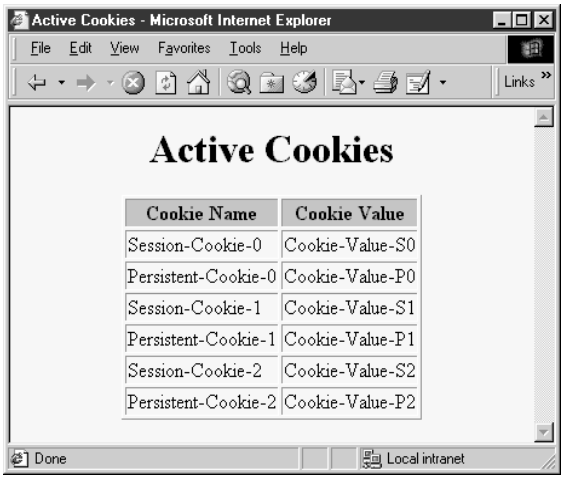


Figure 8–2 Result of visiting the `ShowCookies` servlet within an hour of visiting `SetCookies` in the same browser session.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Figure 8–3 Result of visiting the `ShowCookies` servlet within an hour of visiting `SetCookies` in a different browser session.

8.5 Basic Cookie Utilities

This section presents some simple but useful utilities for dealing with cookies.

Finding Cookies with Specified Names

Listing 8.3 shows a section of `ServletUtilities.java` that simplifies the retrieval of a cookie or cookie value, given a cookie name. The `getCookieValue` method loops through the array of available `Cookie` objects, returning the value of any `Cookie` whose name matches the input. If there is no match, the designated default value is returned. So, for example, my typical approach for dealing with cookies is as follows:

```
Cookie[] cookies = request.getCookies();
String color =
  ServletUtilities.getCookieValue(cookies, "color", "black");
String font =
  ServletUtilities.getCookieValue(cookies, "font", "Arial");
```

The `getCookie` method also loops through the array comparing names, but returns the actual `Cookie` object instead of just the value. That method is for cases when you want to do something with the `Cookie` other than just read its value.

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.3 `ServletUtilities.java`

```
package coreservlets;

import javax.servlet.*;
import javax.servlet.http.*;

public class ServletUtilities {
  // Other methods in this class shown in earlier chapters.

  public static String getCookieValue(Cookie[] cookies,
                                      String cookieName,
                                      String defaultValue) {
    for(int i=0; i<cookies.length; i++) {
      Cookie cookie = cookies[i];
      if (cookieName.equals(cookie.getName()))
        return(cookie.getValue());
    }
    return(defaultValue);
  }

  public static Cookie getCookie(Cookie[] cookies,
                                 String cookieName) {
    for(int i=0; i<cookies.length; i++) {
      Cookie cookie = cookies[i];
      if (cookieName.equals(cookie.getName()))
        return(cookie);
    }
    return(null);
  }
}
```

Creating Long-Lived Cookies

Listing 8.4 shows a small class that you can use instead of `Cookie` if you want your cookie to automatically persist when the client quits the browser. See Listing 8.5 for a servlet that uses this class.

8.6 A Customized Search Engine Interface

Listing 8.5 shows the `CustomizedSearchEngines` servlet, a variation of the `SearchEngines` example previously shown in Section 6.3. Like the `SearchEngines` servlet (see Figure 8–5), the `CustomizedSearchEngines` servlet

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.4 `LongLivedCookie.java`

```
package coreservlets;

import javax.servlet.http.*;

/** Cookie that persists 1 year. Default Cookie doesn't
 *  persist past current session.
 */

public class LongLivedCookie extends Cookie {
  public static final int SECONDS_PER_YEAR = 60*60*24*365;

  public LongLivedCookie(String name, String value) {
    super(name, value);
    setMaxAge(SECONDS_PER_YEAR);
  }
}
```

reads the user choices from the HTML front end and forwards them to the appropriate search engine. In addition, the `CustomizedSearchEngines` servlet returns to the client cookies that store the search values. Then, when the user comes back to the front-end servlet at a later time (even after quitting the browser and restarting), the front-end page is initialized with the values from the previous search.

To accomplish this customization, the front end is dynamically generated instead of coming from a static HTML file (see Listing 8.6 for the source code and Figure 8–4 for the result). The front-end servlet reads the cookie values and uses them for the initial values of the HTML form fields. Note that it would not have been possible for the front end to return the cookies directly to the client. That's because the search selections aren't known until the user interactively fills in the form and submits it, which cannot occur until after the servlet that generated the front end has finished executing.

This example uses the `LongLivedCookie` class, shown in the previous section, for creating a `Cookie` that automatically has a long-term expiration date, instructing the browser to use it beyond the current session.

Second edition of this book: www.coreservlets.com; Sequel: www.moreservlets.com.
Servlet and JSP training courses by book's author: courses.coreservlets.com.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.5 `CustomizedSearchEngines.java`

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;

/** A variation of the SearchEngine servlet that uses
 *  cookies to remember users choices. These values
 *  are then used by the SearchEngineFrontEnd servlet
 *  to initialize the form-based front end.
 */

public class CustomizedSearchEngines extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

    String searchString = request.getParameter("searchString");
    if ((searchString == null) ||
        (searchString.length() == 0)) {
      reportProblem(response, "Missing search string.");
      return;
    }
    Cookie searchStringCookie =
      new LongLivedCookie("searchString", searchString);
    response.addCookie(searchStringCookie);
    // The URLEncoder changes spaces to "+" signs and other
    // non-alphanumeric characters to "%XY", where XY is the
    // hex value of the ASCII (or ISO Latin-1) character.
    // Browsers always URL-encode form values, so the
    // getParameter method decodes automatically. But since
    // we're just passing this on to another server, we need to
    // re-encode it.
    searchString = URLEncoder.encode(searchString);
    String numResults = request.getParameter("numResults");
    if ((numResults == null) ||
        (numResults.equals("0")) ||
        (numResults.length() == 0)) {
      numResults = "10";
    }
    Cookie numResultsCookie =
      new LongLivedCookie("numResults", numResults);
    response.addCookie(numResultsCookie);
    String searchEngine = request.getParameter("searchEngine");
    if (searchEngine == null) {
      reportProblem(response, "Missing search engine name.");
      return;
    }
```

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.5 `CustomizedSearchEngines.java` (continued)

```
    Cookie searchEngineCookie =
      new LongLivedCookie("searchEngine", searchEngine);
    response.addCookie(searchEngineCookie);
    SearchSpec[] commonSpecs = SearchSpec.getCommonSpecs();
    for(int i=0; i<commonSpecs.length; i++) {
      SearchSpec searchSpec = commonSpecs[i];
      if (searchSpec.getName().equals(searchEngine)) {
        String url =
          searchSpec.makeURL(searchString, numResults);
        response.sendRedirect(url);
        return;
      }
    }
    reportProblem(response, "Unrecognized search engine.");
  }

  private void reportProblem(HttpServletResponse response,
                             String message)
      throws IOException {
    response.sendError(response.SC_NOT_FOUND,
                       "<H2>" + message + "</H2>");
  }

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

Listing 8.6 `SearchEnginesFrontEnd.java`

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.net.*;

/** Dynamically generated variation of the
 *  SearchEngines.html front end that uses cookies
 *  to remember a user's preferences.
 */
```

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.6 SearchEnginesFrontEnd.java (continued)

```
public class SearchEnginesFrontEnd extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    Cookie[] cookies = request.getCookies();
    String searchString =
      ServletUtilities.getCookieValue(cookies,
                                      "searchString",
                                      "Java Programming");
    String numResults =
      ServletUtilities.getCookieValue(cookies,
                                      "numResults",
                                      "10");
    String searchEngine =
      ServletUtilities.getCookieValue(cookies,
                                      "searchEngine",
                                      "google");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Searching the Web";
    out.println
      (ServletUtilities.headWithTitle(title) +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +
       "<H1 ALIGN=\"CENTER\">Searching the Web</H1>\n" +
       "\n" +
       "<FORM ACTION=\"/servlet/" +
         "coreservlets.CustomizedSearchEngines\">\n" +
       "<CENTER>\n" +
       "Search String:\n" +
       "<INPUT TYPE=\"TEXT\" NAME=\"searchString\"\n" +
       "       VALUE=\"" + searchString + "\"><BR>\n" +
       "Results to Show Per Page:\n" +
       "<INPUT TYPE=\"TEXT\" NAME=\"numResults\"\n" +
       "       VALUE=" + numResults + " SIZE=3><BR>\n" +
       "<INPUT TYPE=\"RADIO\" NAME=\"searchEngine\"\n" +
       "       VALUE=\"google\"" +
       checked("google", searchEngine) + ">\n" +
       "Google |\n" +
       "<INPUT TYPE=\"RADIO\" NAME=\"searchEngine\"\n" +
                "       VALUE=\"infoseek\"" +
       checked("infoseek", searchEngine) + ">\n" +
       "Infoseek |\n" +
       "<INPUT TYPE=\"RADIO\" NAME=\"searchEngine\"\n" +
       "       VALUE=\"lycos\"" +
       checked("lycos", searchEngine) + ">\n" +
       "Lycos |\n" +
       "<INPUT TYPE=\"RADIO\" NAME=\"searchEngine\"\n" +
       "       VALUE=\"hotbot\"" +
```

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

Listing 8.6 `SearchEnginesFrontEnd.java` (continued)

```
        checked("hotbot", searchEngine) + ">\n" +
        "HotBot\n" +
        "<BR>\n" +
        "<INPUT TYPE=\"SUBMIT\" VALUE=\"Search\">\n" +
        "</CENTER>\n" +
        "</FORM>\n" +
        "\n" +
        "</BODY>\n" +
        "</HTML>\n");
  }

  private String checked(String name1, String name2) {
    if (name1.equals(name2))
      return(" CHECKED");
    else
      return("");
  }
}
```

Figure 8–4 Result of `SearchEnginesFrontEnd` servlet. Whatever options you specify will be the initial choices next time you visit the same servlet.

© Prentice Hall and Sun Microsystems. Personal use only; do not redistribute.

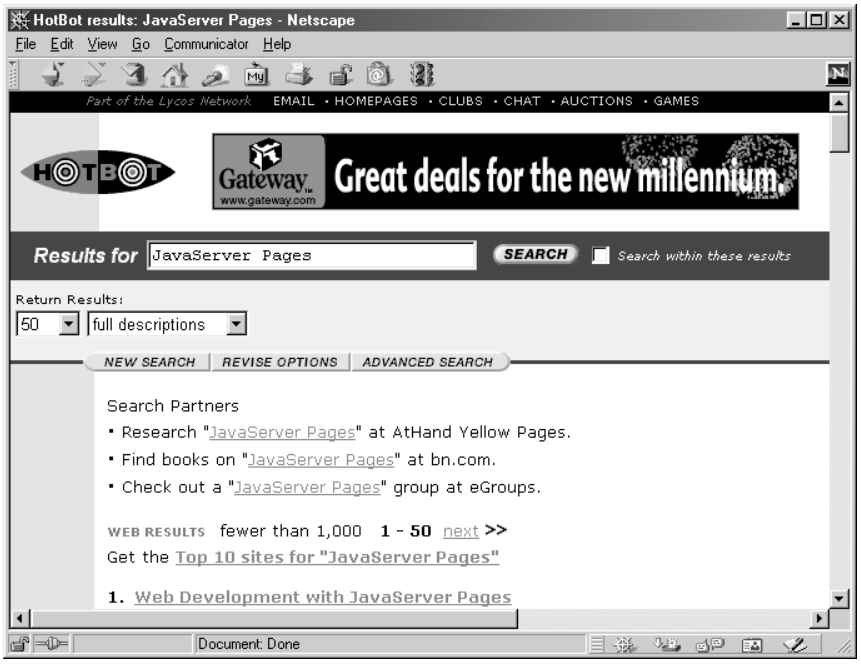


Figure 8–5 Result of `CustomizedSearchEngines` servlet.